School of Electronic Engineering
and Computer Science

Final Year Report

Programme of study:
BSc Computer Science

Project Title:
VisualPromptBuilder – Sentence Formation and Translation for Nonverbal Individuals

Supervisor:
Dr. Haim Dubossarsky

Student Name:
Zaid Kareem Chughtai

Final Year
Undergraduate Project 2023/24

Queen Mary
University of London

Date: 29/04/2024

Acknowledgements

I would like to thank the following people. Without them I would not have been able to complete the research and implementation for this project.

My supervisor Dr. Haim Dubossarsky for continually supporting and guiding to complete this project.

Those who anonymously volunteered to assist me in my service evaluation; the feedback was invaluable.

My family and friends for always being there to care for me during my time on this course, especially during my final year.

Abstract

VisualPromptBuilder aims to develop a communication tool for individuals with speech impairments, utilising pictograms to facilitate sentence formation and speech output in a multitude of languages. This report details the process of research, analysis, designing, implementing, and evaluating a software solution that addresses the need for a simple, intuitive communication aid for nonverbal or limited-speech users. Using pictograms, the tool allows users to construct sentences that are then translated from English to a desired language, then vocalising sentences in both languages via a text-to-speech engine. The project's effectiveness was assessed through both technical performance measures and user feedback, focusing on the ease of use, responsiveness, and overall satisfaction of the target user group. The outcomes suggest a promising approach to enhancing communication for those who rely on assistive technologies.

Contents

List of Figures:

List of Tables:

Chapter 1: **Introduction**

This report presents the development of an assistive web-based tool tailored for nonverbal individuals and those with cognitive disabilities, named VisualPromptBuilder. This application harnesses pictograms, text-to-speech, and translation technologies to provide a comprehensive communication platform. This introduction outlines the purpose and structure of the report, setting the context for my research and design approach.

1.1 Background

The inspiration for this project stems from the communication barriers faced by many nonverbal individuals, including my own family. Current assistive technologies often fall short due to their cost, complexity, or inflexibility. VisualPromptBuilder seeks to fill this gap by offering an intuitive, free solution that leverages modern web technologies for enhanced accessibility and user independence.

1.2 Problem Statement

Due to complex global issues, such as the wars and the energy crisis, there are financial constraints on healthcare services provided in most countries. To support people who are nonverbal or with cognitive disabilities, such as autism and ADHD, there is an urgent need for a universally accessible communication aid that is simple to use, free, open-source and sophisticated enough to handle diverse communication scenarios. The tool aims to transcend the limitations of existing aids by providing a scalable and adaptable solution that integrates seamlessly with modern technology.

1.3 Aim

The project aims to investigate current approaches that are used to help nonverbal users to communicate. It will create a versatile communication aid that breaks down the barriers for nonverbal users. By utilising pictograms and implementing text-to-speech and translation functionalities, VisualPromptBuilder aims to empower users with the ability to articulate sentences and engage in conversations across different languages and platforms. With the advent of advanced technologies such as AI, this project focuses to research and incorporate such related services.

1.4 Objectives

This project's objectives focus on tackling functional and user-experience issues of assistive communication technology. These include:

- To develop an intuitive interface, providing easy navigation and manipulation of pictograms to form sentences.
- To utilise advanced text-to-speech technology capable of sounding out complete and natural sentences made from pictograms.

- To make sure the application is responsive and works on platforms such as mobiles and web browsers, giving accessibility to the users.
- To include features that allow for sentence translation, speech output and support for multiple languages.
- The application should be easy to extend and maintain.

The design goals are to address immediate needs of communication for its users, while building an opensource, scalable and adaptable framework able to evolve with technological advance and user feedback.

1.5 Research Questions

The research questions for the project are crafted to guide the investigation and development of the tool, focusing on aspects critical to its success and usability. These questions include:

- How can pictogram-based communication tools be optimised for ease of use and effectiveness for nonverbal users?
- What are the technical requirements for integrating text-to-speech technology that accurately and naturally vocalises sentences formed from pictograms?
- In what ways can the UI be designed to accommodate users with varying abilities and preferences, enhancing accessibility and customisation?
- What are the challenges associated with developing a cross-platform communication tool, and how can these be effectively addressed?
- How can multilingual support be implemented in a way that is both efficient and useful for users speaking different languages?
- How is this application going to be an improvement, compared to existing solutions?

These questions aim to explore both the theoretical and practical aspects of developing a communication tool that is not only functional but also user centred. They serve to identify the key areas of focus in the project's development and evaluation stages.

1.6 Project Evolution

This project started from a need to provide a socially positive solution that caters to users with specialised needs. Initially the concept was based on using existing software for data extraction to develop an AI model, but due to challenges mentioned below, this transformed into something more innovative.

1.6.1 Initial Concept and Challenges

It started with the idea to use Widgit’s InPrint 3 software (Refer to Section 2.3 for more details) to create a dataset of sentences formed by pictograms, which could then be used to train an AI model to convert sets of pictograms directly into cohesive sentences. InPrint's functionality to overlay pictograms above key words in sentences seemed promising; it simplified sentences by omitting semantically unnecessary words, thus

providing a clear structure of essential cues in communication. But a significant hurdle was encountered: InPrint allowed only the saving of these frames as images rather than selectable text. This limitation was compounded by the inability to export data without severe restrictions, such as watermarking in the free trial version, making it impractical to extract the necessary textual data for AI training.

1.6.2 Pivoting the Project Focus

Given the constraints with data extraction from InPrint (free edition) and the impending project deadlines, a pivot was necessary. The focus shifted from developing an AI model for sentence reconstruction to developing an application that would be open-source and free, with enhanced functionality. The revised project, VisualPromptBuilder, now aimed to enable users to select pictograms directly and form sentences in a communication bar, reflecting a more interactive and user-driven approach.

1.6.3 Development of VisualPromptBuilder

VisualPromptBuilder was designed with a grid of clickable pictograms and categorisation folders to simplify the selection process for users. This design supports intuitive navigation and sentence formation, essential for nonverbal users' effective communication. The communication bar, where selected pictograms appear sequentially and can be adjusted with undo and bin options, allows for dynamic sentence construction and modification, a crucial feature for tailoring communication in real-time.

Additionally, recognising the multilingual needs of various users, the application incorporated a feature to translate sentences into multiple languages and vocalise them using text-to-speech technology. This not only enhanced the application's inclusivity but also its applicability in diverse linguistic and cultural contexts.

1.6.4 Anticipated User Testing and Ethical Considerations

The plans for the project include user testing to refine and optimise the interface and functionalities based on real-world usage. Ethical considerations, particularly concerning user privacy and data security, are paramount, given the sensitive nature of the application’s target user group. Such testing will adhere to ethical standards, ensuring that the enhancements are both user-centric and socially responsible.

1.7 Report Structure

This report is structured to clearly outline the progression from project inception to evaluation, organised as follows:

Chapter 1: Introduction - Outlines the background, aims, objectives, and foundational questions that guide the project.

Chapter 2: Literature Review - Reviews existing research and technologies in assistive communication tools, identifying gaps and contextualising this project.

Chapter 3: Solution Analysis, Design, and Implementation - Details the architectural design, system features, and implementation strategies of VisualPromptBuilder.

Chapter 4: Development Process - Describes the development methodologies, coding practices, and challenges encountered during the project.

Chapter 5: Testing and Evaluation - Discusses the testing methods employed, presents evaluation results, and includes feedback from user testing.

Chapter 6: Discussion - Analyses the results in relation to the project's objectives and existing technologies, emphasising successes and areas for improvement.

Chapter 7: Conclusion and Future Work - Summarises the project's achievements and outlines potential future enhancements and research directions.

Chapter 8: References - Lists all sources cited throughout the report, including applications accessed.

Appendices - Includes additional materials such as a glossary, code snippets, user guides, and design schematics that support the report's documentation.

Chapter 2: **Literature Review**

The literature review explores existing research and practical applications within the field of assistive communication technologies, with a focus on tools designed for nonverbal and limited-speech users. This section delves into the evolution of communication aids, from simple symbol-based systems to sophisticated software that integrates text-to-speech functionalities. It highlights the significance of pictograms in enhancing communication for individuals with speech impairments, examining their effectiveness in conveying complex ideas succinctly and intuitively.

2.1 Design Considerations for Assistive Tools

Zashchirinskaia, (2020) research findings support the urgent need for tools to help nonverbal users, especially children. These users are generally vulnerable, and many are dependent on a carer. Morris et al. (2016) shared that in 2012, 10% of US Adult population reported a communication disability. Zashchirinskaia, (2020) work highlights for these users, there is a need for solution analysis and design to consider factors such as behavioural, cognitive and emotional states. Plomin et al. (2002) work supports the association between such users and some behavioural issues. This aligns with the Design Thinking approach, as it includes emphasising with users, iteratively building a solution (with the user) and enhancing the end-user experience. Carlgren et al., (2016) research found that Design Thinking approaches need to be tailored to the target solution and needs, with some focusing more on problem framing and other usage focusing on visualisation of the solution. As such individuals, may have varying cognitive abilities, key analysis and design discussions will need to be with the carer.

Okamoto et al., (2022) research tracked eye gaze positions of nonverbal individuals using eye tracking and video analysis. Okamoto et al., (2022) found that nonverbal individuals are more likely to have:

1. Erratic eye movements
2. Find following instructions difficult
3. Very short periods of interest and this is focused on one specific object

Work by Okamoto et al. (2022) supports need for the solution to be very simple to use, as the user requirements should focus more on accessibility than attractiveness. This supports the work by Motti et al. (2015), which highlighted unique challenges for nonverbal and elderly users, such as using drag and drop interfaces.

Carlgren et al. (2016) and Peters et al. (2020) to utilise design thinking process to understand empathetic behaviour. The Responsible design process framework by Peters et al. (2020) is useful, however, it needs to be slightly adapted, as the project users cannot directly be interviewed about their wellbeing. Carlgren et al. (2016) did not specific which aspects would be suitable for nonverbal individuals, however the following based on Shinohara et al. (2016) seem to be suitable; focusing on the visualisation (of the solution demo), iterative nature, as nonverbal users will not easily be able to provide significant feedback and user focus (understanding the specific needs). Li and Zhao. (2015) shared the symbols are a useful way to express and communicate for nonverbal users and when discussing in non-English speaking countries.

2.2 Existing Solution Comparison

Feature	InPrint	PicTalk	Boardmaker
Interface and Usability	Offers pre-designed templates and drag-and-drop interface.	User-friendly interface, customisable with a large symbol library for personal expression.	Customisable templates, accessible design tailored for educational settings.
Symbol Library	Over 20,000 Widgit Symbols, highly customisable.	More than 15,000 symbols and icons, allows users to create personalised communication boards.	Extensive symbol library with PECS (Picture Exchange Communication System) symbols included.
Multilingual Support	Strong multilingual support and text-to-speech capabilities.	Multilingual support with customisable translations, text-to-speech with adjustable speed and pitch.	Multilingual text-to-speech support, though specific language options vary.
Educational and Therapeutic Use	Widely used in educational settings, supports learning and independence.	Suitable for children and adults with communication challenges, aids in expressing thoughts and needs.	Strong educational focus, widely used in special education for curriculum development.
Technology Integration and Platform Support	Desktop application with network-friendly features.	Mobile application, available on iOS, focuses on ease of use across personal devices.	Available on multiple platforms, integrates well with educational technologies.

Table 1 – AAC Systems Comparison

2.3 Inprint 3

InPrint (specifically InPrint 3) from Widgit Software is a tool designed to facilitate communication through the use of symbols. This software is useful for individuals with cognitive disabilities, providing a user-friendly platform to create visual aids like communication boards, learning materials, and other resources. Li and Zhao. (2015) shared that communication using symbols is also beneficial for international business.

2.3.1 Interface and Usability

InPrint is designed with educators, therapists, and caregivers in mind who require quick and easy tools to support communication. It features an intuitive drag-and-drop system that simplifies the process of producing custom communication aids. InPrint includes a variety of pre-designed templates that users can adapt to meet their specific needs. It also includes functionality to automatically generate symbols from typed in sentences. These symbol sets are generally used by the caretakers of those with cognitive disabilities.

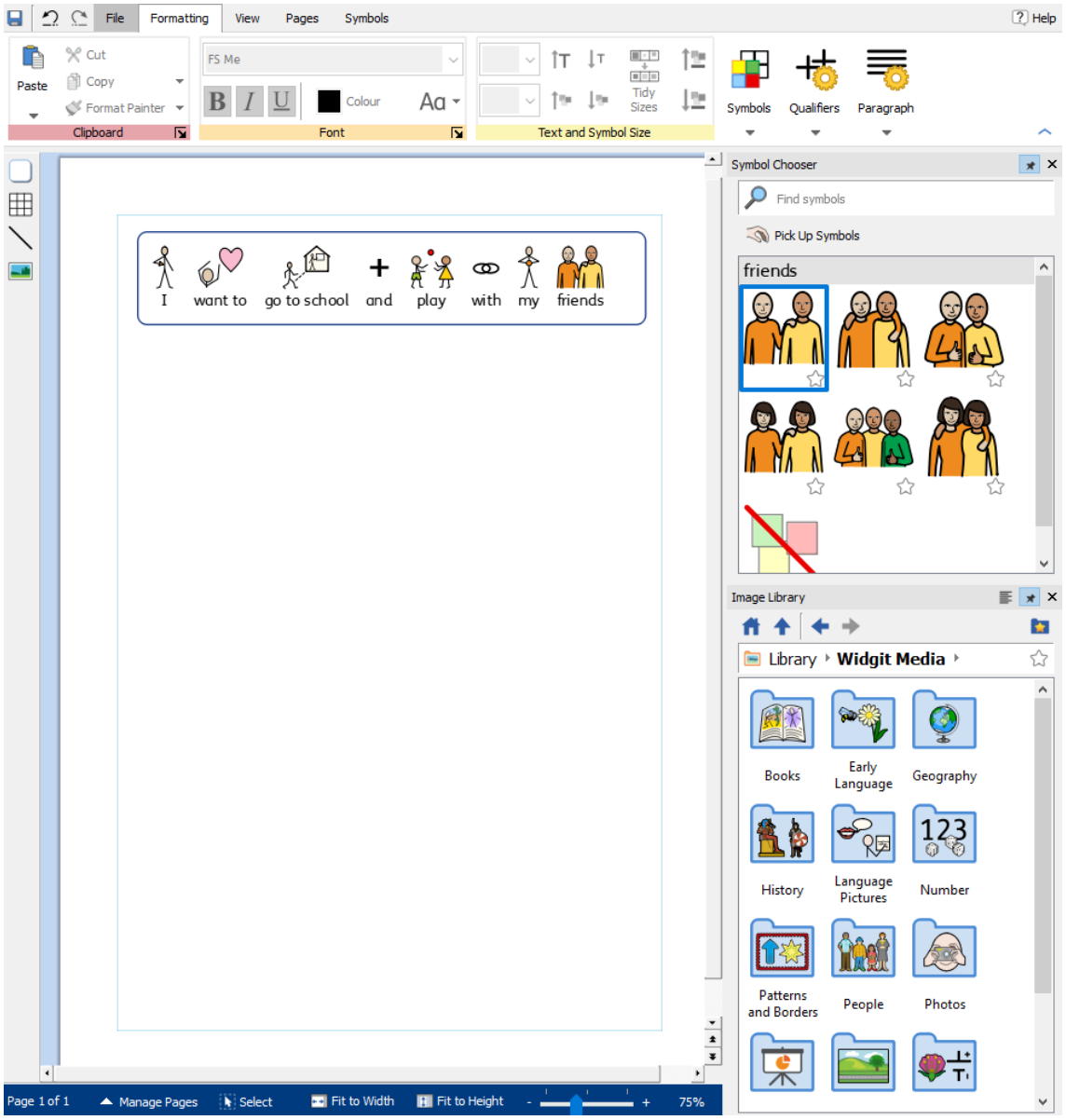

Figure 1 – InPrint Interface

2.3.2 Symbol Library and Customisation

The symbol library in InPrint is extensive, with 20,000+ Widgit Symbols covering a wide range of needs. Users can also add their own custom pictograms to be used in the app, allowing for comprehensive and flexible communication.

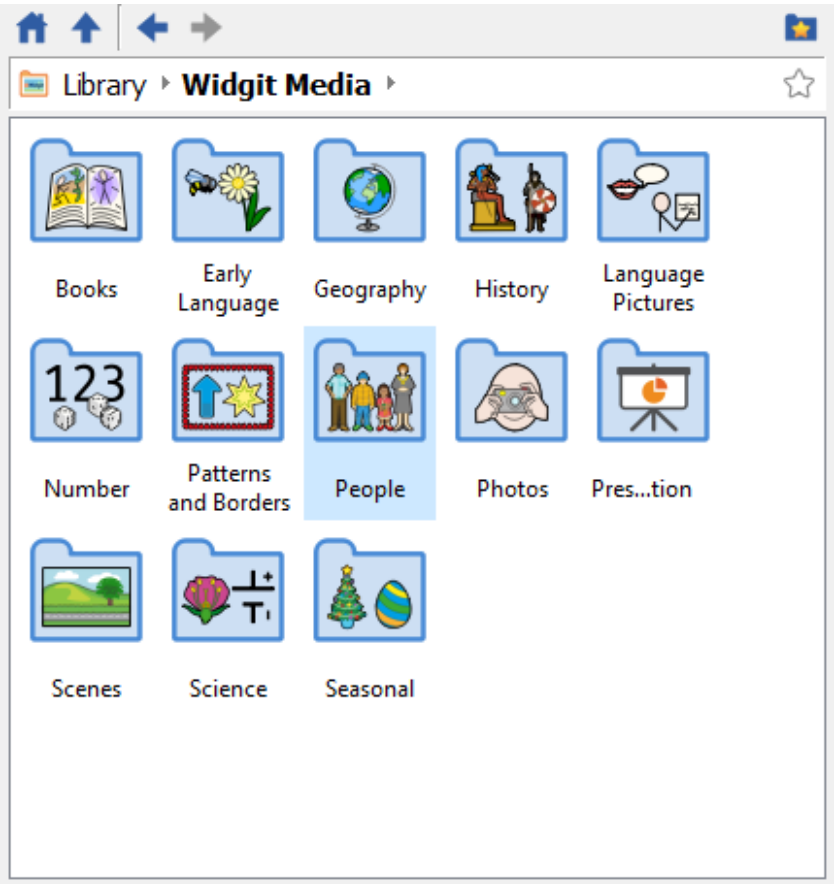

Figure 2 - InPrint Symbol/Folder Library

Figure 3 – Contents of InPrint Folder

2.3.3 Multilingual Support and Text-to-Speech

InPrint lacks multilingual support and text-to-speech capabilities, which are crucial for non-English speakers and those who benefit from auditory learning. This limitation restricts the software's utility in diverse linguistic settings and reduces its effectiveness as a comprehensive communication aid.

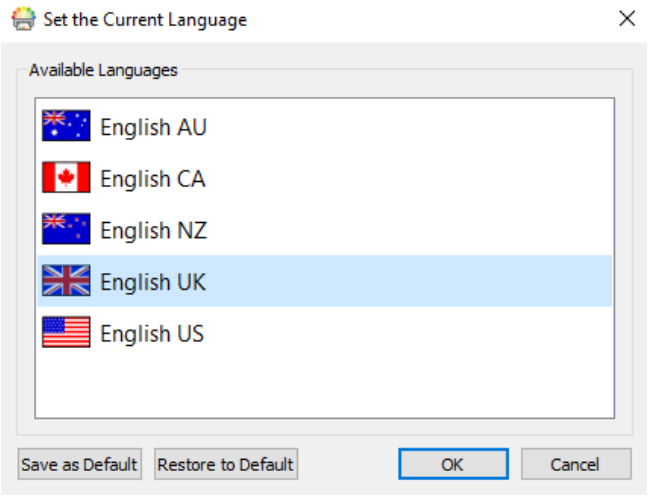

Figure 4 – InPrint's Lack of Linguistic Support

2.3.4 Educational and Therapeutic Use

InPrint supports nonverbal student independence by facilitating the creation of materials that are tailored to the learning speeds and styles of various users.

2.3.5 Technology Integration and Platform Support

InPrint requires desktop installation, which could limit accessibility for users who rely on mobile devices or prefer web-based applications. This stresses the need for broader platform support to accommodate the varied tech environments of its users.

2.3.6 Comparative Analysis and Project Pivoting

2.3.6.1 Initial Concept and Challenges

The initial phase of VisualPromptBuilder explored the possibility of using InPrint to create a dataset of sentences represented by pictograms. This was intended to train an AI to translate pictogram sets directly into text. Unfortunately, InPrint (free edition) only allowed saving these frames as images, not as selectable or exportable text. This limitation, along with data export restrictions like watermarking in the free trial version, made it impractical for gathering the necessary data.

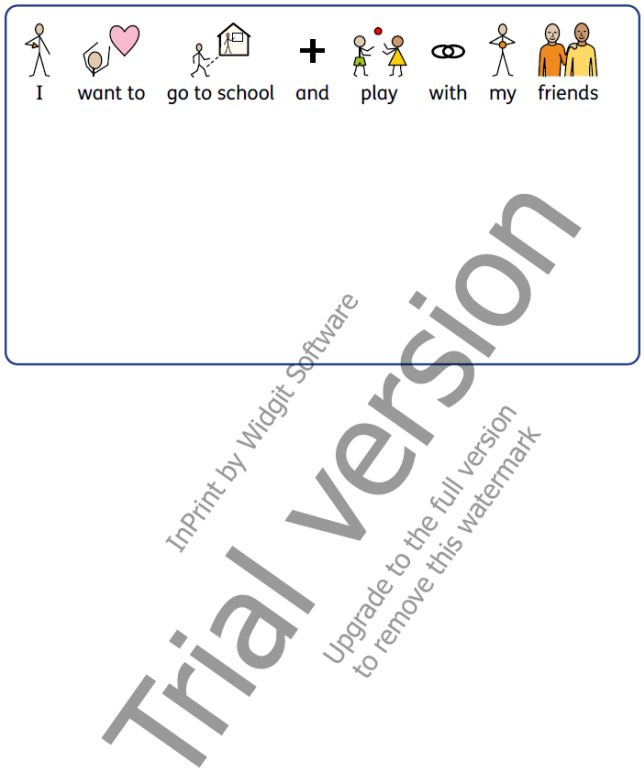

Figure 5 – InPrint PDF Export with Watermark

2.3.6.2 Pivoting the Project Focus

Considering that InPrint had limited functionality in the trial edition, this was seen as an opportunity to help users who cannot use the paid version or related solutions. Another aspect; Motti et al.'s (2015) research found some nonverbal users will struggle with drag and drop interfaces, so the decision was to utilise a simplified user experience in the project. VisualPromptBuilder was redesigned to allow users to directly select pictograms and form sentences in a communication bar, which is more interactive and responsive to user input.

2.3.6.3 Development and Anticipated Enhancements

VisualPromptBuilder features a grid of clickable pictograms, categorisation folders, and a communication bar for sentence formation. Anticipating the needs of a diverse user base, the application includes translation and text-to-speech features, addressing the key limitations of InPrint and enhancing its applicability in multilingual contexts.

2.4 Pictalk.org

An open-source application which I drew most of my project's ideas and inspiration from is Pictalk.org, a webpage that achieves similar functionality to my project. It is also an innovative tool designed to assist nonverbal individuals in communication.

2.4.1 Interface and Usability

Both Pictalk.org and VisualPromptBuilder prioritise an intuitive user interface, which is crucial for users with varying levels of technical skills. The interfaces are designed to ensure that users can navigate the systems effortlessly, with clear labels and responsive design layouts that are accessible on multiple devices. This ease of use extends to the visual presentation of choices, where both systems utilise a grid layout that displays

pictograms in an organised manner, making it simple for users to construct sentences quickly.

2.4.2 Symbol Library and Customisation

Pictalk.org offers customisation options that allow users to add their pictograms or modify existing ones, enhancing personalisation.

2.4.3 Core Functionality and User Engagement

Both systems are quite similar - users select desired symbols that are then concatenated into sentences, displayed in a bar visually on the screen. This method supports users in making coherent messages that can be understood by others, promoting greater independence in communication.

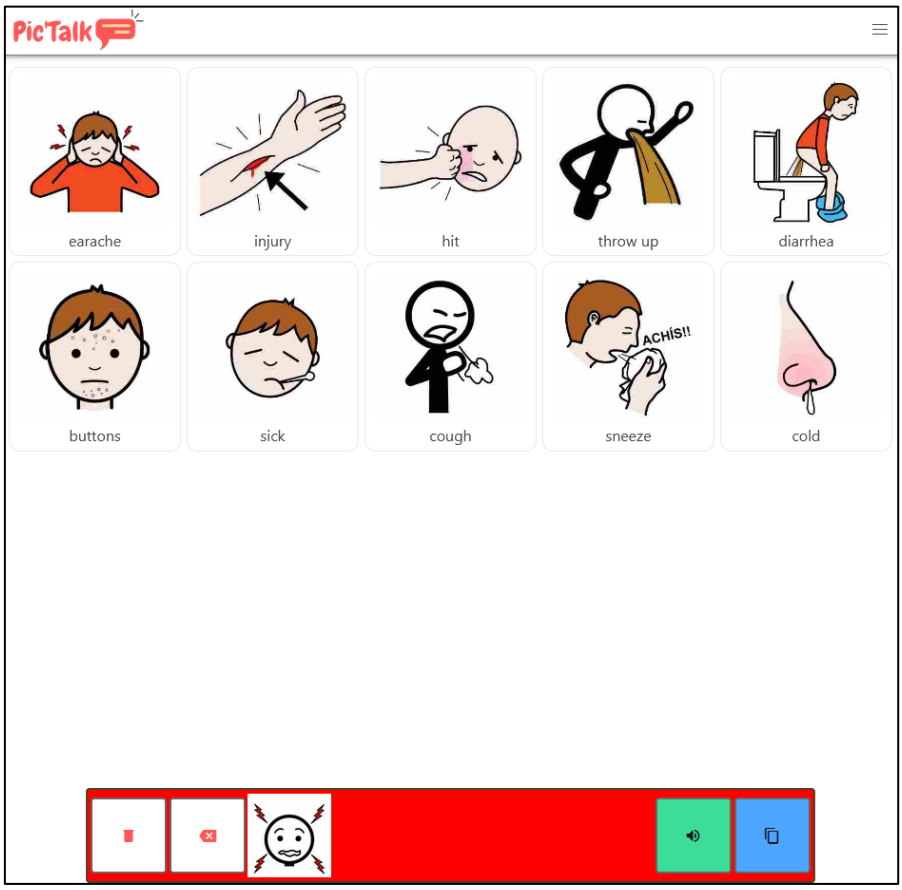

Figure 6 - Pictalk.org UI and Features

2.4.4 Unique Features – Translation Functionality

VisualPromptBuilder introduces a translation functionality, extending the utility of the system by enabling the translation of constructed pictogram sentences into multiple languages. This capability is beneficial in multilingual environments or where caregivers and the user do not share a common language, thus broadening the scope of communication and enhancing inclusivity.

2.4.5 Conclusion

While both Pictalk.org and VisualPromptBuilder aid communication for nonverbal individuals through pictograms, the addition of a translation functionality in VisualPromptBuilder allows for wider usability across different linguistic backgrounds, fostering a more inclusive approach to assistive communication technologies.

2.5 Assistive Technologies for Cognitive Disabilities

In the context of this project, this literature review draws upon the comprehensive insights provided by Scherer et al. (2005) regarding assistive technologies (AT) tailored for individuals with cognitive disabilities. Their research underlines the necessity for AT to be adaptable and personalised, reflecting the core design philosophy of my application.

2.5.1 Adaptation and Personalisation in Assistive Technology

VisualPromptBuilder's development has been informed by the paper's advocacy for AT personalisation. My approach echoes this sentiment, focusing on a user-centric design that adapts to someone's environment, thereby facilitating better communication.

2.5.2 Multidisciplinary Approaches

VisualPromptBuilder's design strategy exemplifies the multidisciplinary approach to assistive technology development. By integrating principles from various disciplines such as Design Thinking, UI/UX design, software engineering, and cognitive psychology, the application is not just technically proficient but also deeply attuned to the practical realities of the users.

2.5.2.1 Importance of Design Thinking and Individual Assessments

To understand disabled user needs, Shinohara et al. (2016) support that design thinking should be used. This includes individual assessments and interviews (uncovering mental models and pain points) during the design thinking 'Empathise' stage. Peters et al. (2020) goes further and states that wellbeing and ethical impact analysis should be part of the process.

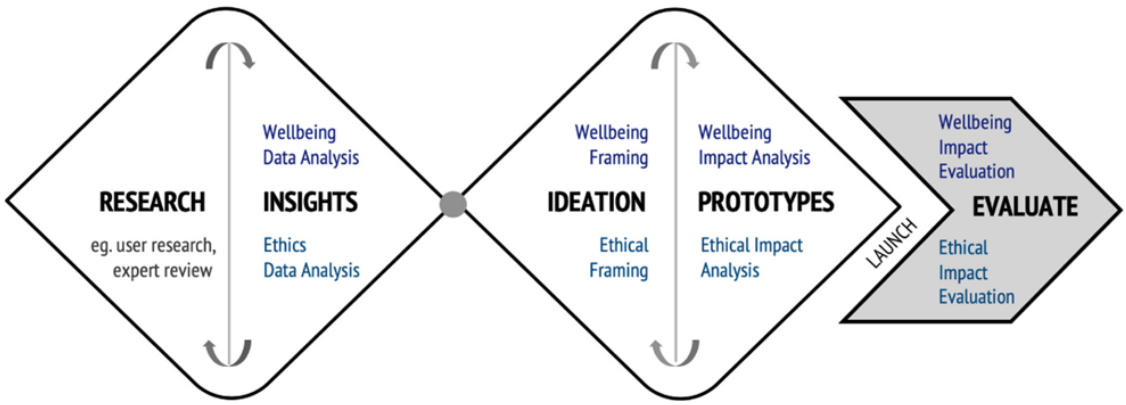

Figure 7 - Responsible Design Process Framework

VisualPromptBuilder's application design incorporates the value of individual assessments highlighted in the paper, ensuring that VisualPromptBuilder is versatile and meets the varied needs of the target demographic.

The integration of the Person-Environment-Occupation (PEO) model into VisualPromptBuilder enhances its capability to support users effectively in their day-to-day tasks, reaffirming the commitment to creating a tool that is as practical as it is innovative.

2.5.2.2 Collaborative Development

The project embraces the collaborative development approach, actively incorporating user feedback into iterative updates for continuous improvement of functionality and user experience.

2.5.3 Transitioning to High-Tech Solutions

Reflective of the paper's discussion, VisualPromptBuilder represents a paradigm shift towards leveraging high-tech solutions in AT. It balances advanced technology integration with a commitment to maintain simplicity, ensuring the platform remains accessible to all cognitive abilities.

2.5.3.1 Technology Complexity and User Experience

The application embodies the paper's narrative on managing technological complexity while preserving an intuitive user experience, facilitating a frictionless interaction for users.

2.5.3.2 Long-Term Engagement

Incorporating the concept of long-term engagement, VisualPromptBuilder is designed for enduring use, supporting users as their needs and technologies evolve.

2.5.4 Training and Support in AT Usage

The project considers the essential role of training and support in effective AT usage as asserted by Scherer et al., (2005) which is why VisualPromptBuilder includes comprehensive guides and support systems to aid both new and experienced users.

2.5.5 Conclusion and Future Directions

In conclusion, the insights from Scherer et al.'s (2005) paper significantly influence the project. Their emphasis on personalisation, a multidisciplinary approach, and user-centred design is woven into the fabric of the application, shaping it to be a malleable extension of the user. This ensures that my project not only serves as a communication tool but also enriches users' interactions with the world.

2.6 Integration into a PECS Learning Environment

PECS is an evidence-based practice designed primarily for individuals with autism and related developmental disabilities to help them acquire functional communication skills. The system typically progresses through several phases, from simple picture exchanges to forming sentences and commenting.

Raharjo et al., (2022) research found that nonverbal communication plays a vital role in the lives of individuals with autism spectrum disorder, impacting their social interactions, overall communication skills and employment opportunities.

2.6.1 Similarities with VisualPromptBuilder

Both PECS and VisualPromptBuilder utilise visual symbols to aid communication, making the basic methodology compatible. VisualPromptBuilder could naturally extend the capabilities of PECS by offering more sophisticated communication tools through its translation functionality, which could support non-native speakers or multilingual environments.

2.6.2 Phases Suitable for Integration

2.6.2.1 Phase III and IV

These phases involve sentence structure and response to questions, where VisualPromptBuilder can be particularly useful. By integrating VisualPromptBuilder into these phases, users could not only form basic requests but also translate them into multiple languages, enhancing communication possibilities with diverse audiences.

2.6.3 Training Programs

Implementing VisualPromptBuilder within a PECS framework would require tailored training programs for educators and caregivers. These programs would need to include:

- Understanding the software's interface and features.
- How to effectively integrate translation features into daily communication tasks.
- When dealing with multiple languages, understanding cultural nuances is crucial.

2.6.4 Simpler Solutions

While PECS is relatively low-tech - involving physical cards for communication - VisualPromptBuilder offers a digital, portable alternative that could simplify logistics and enhance the dynamism of communication through instant translation.

2.7 ChatGPT vs Bard for Sentence Generation

The evolution of Language Model Multimodal Systems (LLMs), specifically in their capacity to construct meaningful sentences from sets of cues—referred to as threads—has profound implications for their applicability in assistive technologies.

2.7.1 Prompt Engineering Evolution

Through iterative prompt engineering, I repetitively tested Bard (now Gemini) and ChatGPT-4 to generate coherent sentences based on given threads, which are essentially semantically incorrect sentences comprised of textual keywords corresponding to pictograms. (Refer to Appendix D)

2.7.1.1 Initial Prompt Interpretation

This involved the LLMs receiving a series of cues, such as `{wishes} {me} {shorts} {sun}`, with the expectation of forming a coherent sentence like, "I desire to wear shorts because of the sunny weather." These cues acted as text replacements for pictograms. Initial observations suggested: while the cues were effectively integrated into sentences, the output was confined to the input without considering external contexts.

2.7.1.2 Iterated Prompt Identification

The models displayed the capability to refine sentence construction and even generate complex sentences with multiple cues. The feedback loop, an integral part of this iteration, allowed for the improvement of accuracy in constructing sentences relevant to the user's intent.

2.7.2 Comparative Observations of ChatGPT and Bard

ChatGPT's responses tended to be concise, which aligns with the desired simplicity for users with cognitive disabilities. Conversely, Bard usually crafted elaborate sentences in

rich descriptive detail, which, while creatively valuable, may not be as suitable for the intended straightforward communication.

2.7.3 Improvement Trajectories

As the iterations progressed, the prompts were refined to balance simplicity with engaging content. Iteration 5, for example, demonstrated increased flexibility by incorporating user feedback to explore narrative variations. It highlighted the importance of a model's ability to adapt narratives based on subtle cue changes, enhancing communicative versatility.

2.7.4 Application to VisualPromptBuilder

The insights gleaned from the prompt engineering research are directly applicable to VisualPromptBuilder. The ability of LLMs to generate sentences from pictogram-based cues can significantly augment the tool's effectiveness. The refinement of prompts and the iterative learning shown in the research underscore the potential of integrating such technology to improve the user experience for nonverbal individuals.

Chapter 3: **Solution Analysis, Design and Implementation**

3.1 Analysis and Design

Integrating human-centric design principles ensures human interaction, fostering trust and user agency. Design Thinking has the following processes:

- Empathy
- Define the Problem
- Ideate Solutions
- Prototype and Test
- Implement and Iterate

The empathy process requires understanding of user fears, gains, needs, and wants plus what the user sees and hears. Without directly working with the end users, as they are nonverbal or have limited cognitive abilities, a trained expert in PECS was interviewed. Based on the interview and research conducted, there is a lack a support for translational services in related applications.

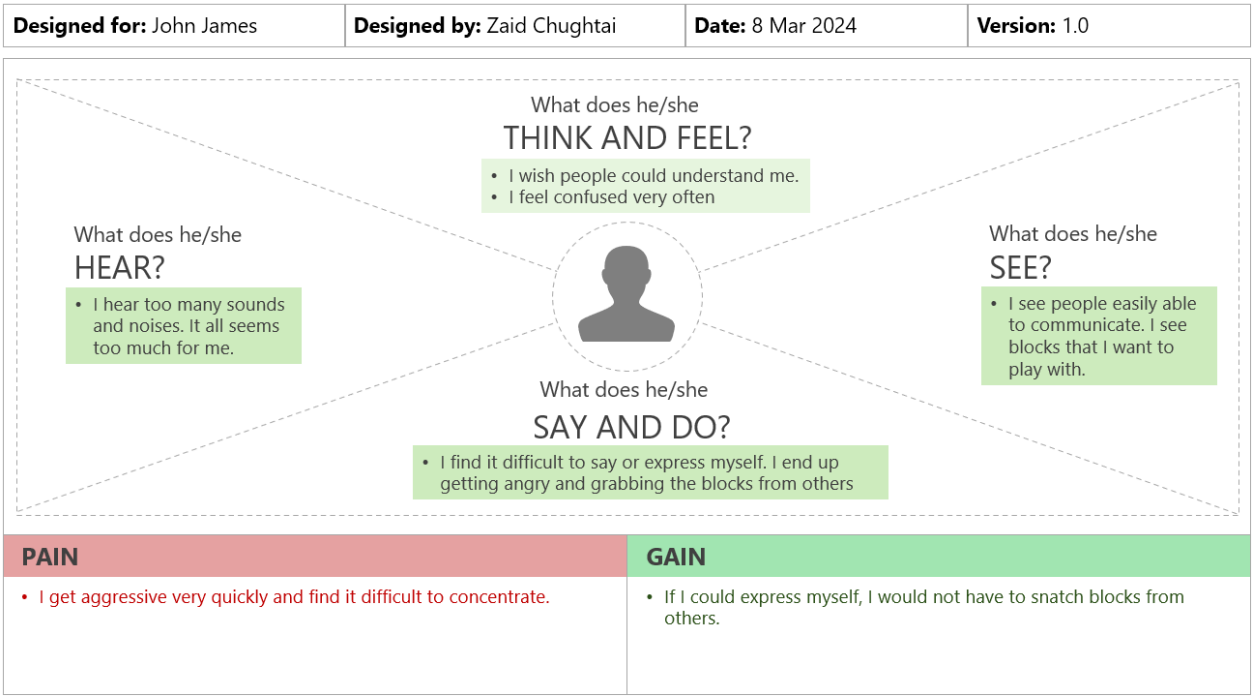

Figure 8 – Empathy Map for a nonverbal persona

The Empathy map was based on the information from the trained expert. This was due to the challenge of producing a solution for nonverbal users. The Empathy map supported the findings from Plomin et al. (2002), which found nonverbal users are associated with some behavioural issues and can cause disruptions.

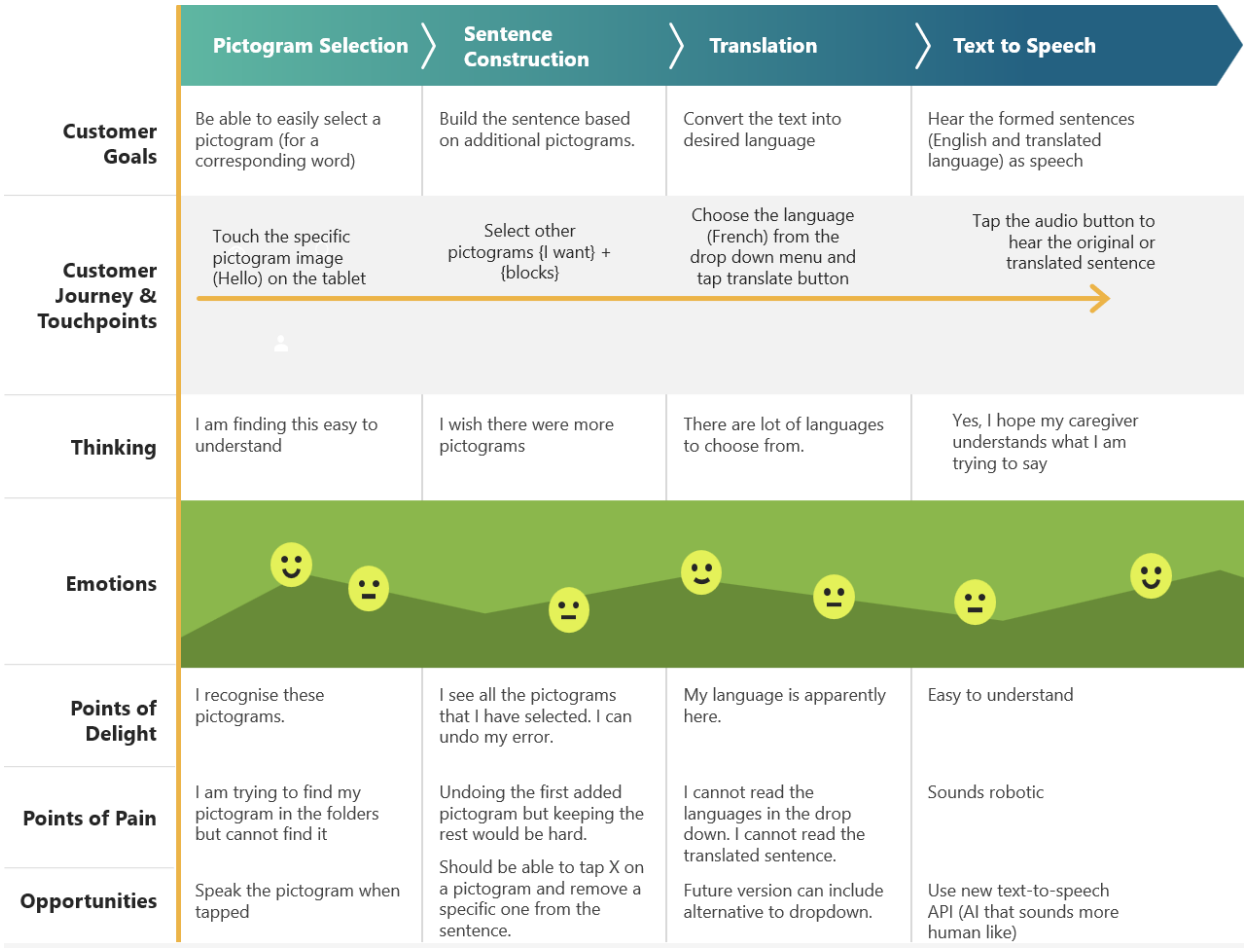

Figure 9 - User Journey for a nonverbal persona

The User Journey Map provides an insight into a particular user, as they go through the application. It is different from a process flow, as the user feelings and pain points are also captured. Based on the interview, the user journey map was produced, highlighting key touchpoints that are either a concern or delight the user.

Based on the research, analysis, and the interview (with the expert) the following functional and non-functional requirements were determined:

3.1.1 Functionality Requirements

The functional requirements are listed to address the needs of the users and will collectively become the features of the solution.

3.1.1.1 Core Functionality

- Pictogram Selection
- Ease of browsing and selecting pictograms from different categories.
- Sentence Building
- Correct recognition and ordering of selected pictograms.
- Undo last added pictogram
- Be able to clear the sentence generated and start again

3.1.1.2 Text Translation

- Accuracy of translating pictogram meaning into natural language text.
- Support for translation into multiple languages
- User options for reviewing or editing translated text.

3.1.1.3 Text-to-Speech

- Natural-sounding speech generation from translated text.
- Accuracy of speech output reflecting sentence structure and meaning.

3.1.1.4 Output Options

- Clear display of translated text on the screen.
- Clear audio of speech
- Application functionality without an internet connection.
- Local storage of pictograms and necessary data for offline use.

3.1.1.5 Error Handling

- Clear and informative error messages in case of issues.
- Graceful recovery from unexpected errors or crashes.

3.1.2 Non-Functionality requirements

3.1.2.1 Performance

- Less than 5 second loading and response on different devices.
- Absence of noticeable lag or performance bottlenecks.

3.1.2.2 Stability

- No unexpected crashes or freezes during use.
- Reliable navigation and action execution.

3.1.2.3 Security

- Should comply with Open Web Application Security Project (OWASP, 2021) Top 10

Rank	Category	Description
1	Broken Access Control	Improper user access restrictions.
2	Cryptographic Failures	Issues in encryption leading to data exposure.
3	Injection	Flaws like SQL injection from untrusted data.
4	Insecure Design	Lack of foundational security design.
5	Security Misconfiguration	Incorrect or incomplete security settings.

6	Vulnerable Components	Use of outdated or vulnerable software parts.
7	Auth Failures	Weaknesses in verifying user identity.
8	Data Integrity Failures	Untrusted code/data affecting software integrity.
9	Logging Failures	Inadequate logging of security events.
10	Server-Side Request Forgery	Server tricked into making unintended requests.

Table 2 –Top 10 Security Risks (OWASP, 2021)

3.1.2.4 Usability and Design Requirements

Based on the design principles from Pernice et al., (2001) and research, the following usability and design requirements will be implemented:

Design Requirement	Description
Purpose Clarity	Users must immediately grasp the application's purpose at first glance.
Feature Accessibility	Critical features should be easily locatable and identifiable.
Intuitive Design	The layout and navigation must be intuitive, promoting ease of use.
Quick Mastery	Users should quickly learn and master the application's core functions.
Feature Usability	All functionalities should be straightforward, avoiding user confusion.
Satisfaction & Recommendation	The application should satisfy users to the extent that they would recommend it to others.
Effortless Pictogram Use	Finding and using pictograms or main interaction elements should be effortless and intuitive.
Logical Organisation	The categorisation and structure of content and features within the application should make sense to users.

Table 3 – Design and Usability Requirements

3.1.3 Accessibility

A key set of requirements are related to making the solution accessible to users with a range of abilities, including those with cognitive disabilities. This is based on the requirements from Web Content Accessibility Guidelines (WCAG).

Category	Requirement	Description
General	WCAG Conformance	Aim for WCAG 2.1 Level A & AA conformance for basic accessibility.
	Documentation	Provide clear, concise, and accessible user documentation

Content	Text Alternatives	Ensure all non-text content (images, pictograms) has clear and concise alternative text descriptions.
	Colour Contrast	Verify sufficient colour contrast between text and background for low vision users.
	Keyboard Accessibility	All functionalities should be accessible using a keyboard without relying solely on a mouse.
	Focus Management	Keyboard focus should be clear and predictable when navigating the application.
	Error Identification & Correction	Provide clear instructions and mechanisms for users to understand and correct errors.
User Interface	Resize Text	Allow users to adjust text size to their preferences for better readability.
	Reflow	Ensure content reflows and resizes properly on different screen sizes and resolutions for optimal user experience (mobile, tablets, desktops).
Additional Considerations	Screen Reader Compatibility	Test the application with screen readers to ensure accurate content presentation and smooth navigation.
	User Testing	Conduct user testing with individuals with disabilities to identify and address any accessibility barriers.

Table 4 – WCAG Compliance Requirements

After the Empathy and Defining the Problem, the next step is to Ideate Solutions. This included different design solutions that address the challenges. One design solution was to produce a mobile app, however, due to the challenges faced by nonverbal and cognitive disadvantaged people, this was obviously not a suitable solution. At this stage, the solution should be easily accessible on a broad spectrum of devices (such as tablets or computers). Peters et al. (2020) framework (based on design thinking) aims to create solutions aligning with human capabilities and improve the overall user journey. Peters et al. (2020) suggest that in the future, usage data can be used (with permission) to assess users' physiological responses to technology interactions. The next section explains the design of the prototype of the solution. This prototype was used to capture feedback, as part of the testing.

3.1.4 Mock Wireframe

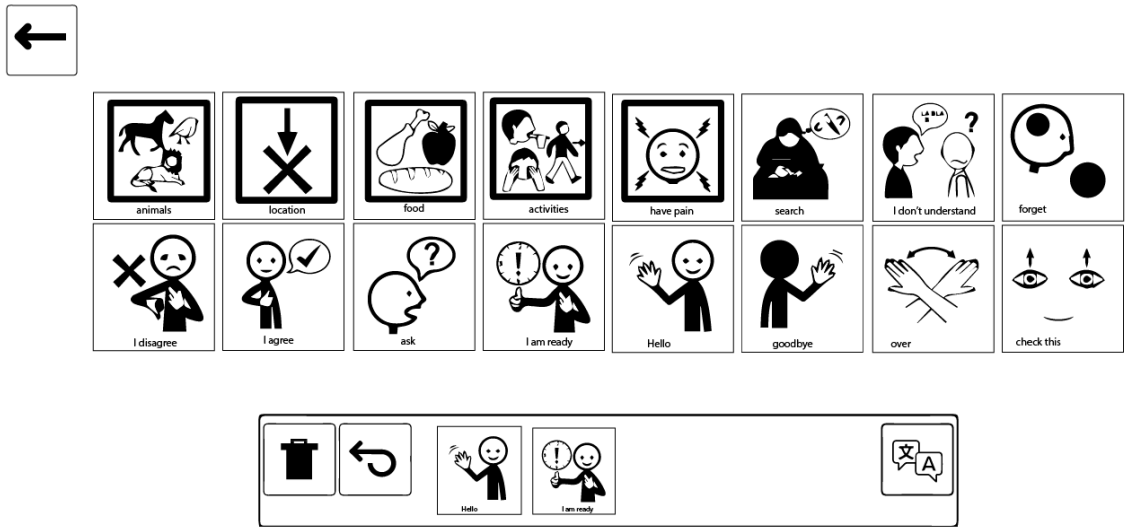

Figure 10 – Prototype Wireframe Layout

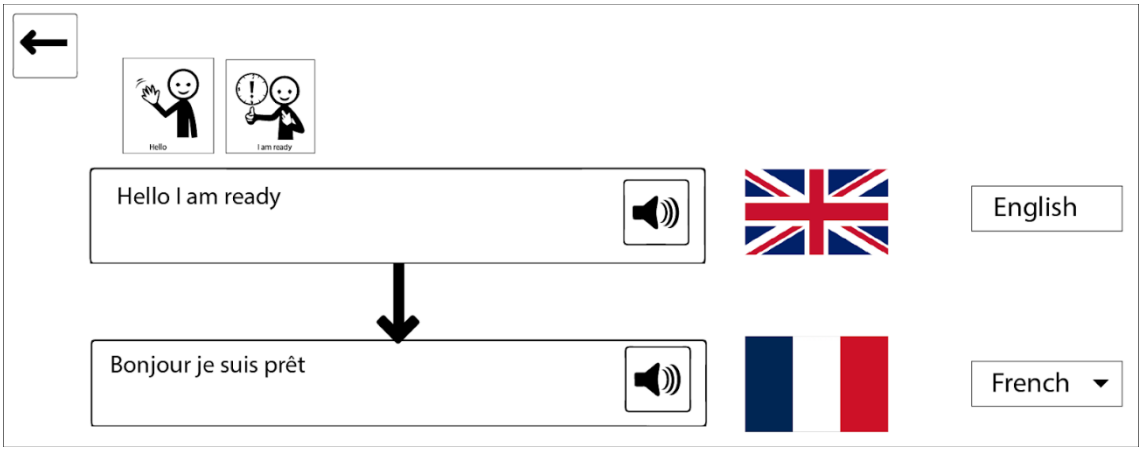

Figure 11 - Prototype Wireframe: Translation & Vocalisation

The mock wireframes were used to capture feedback about the interface.

3.2 System Architecture

3.2.1 Technologies Considered for Prototype

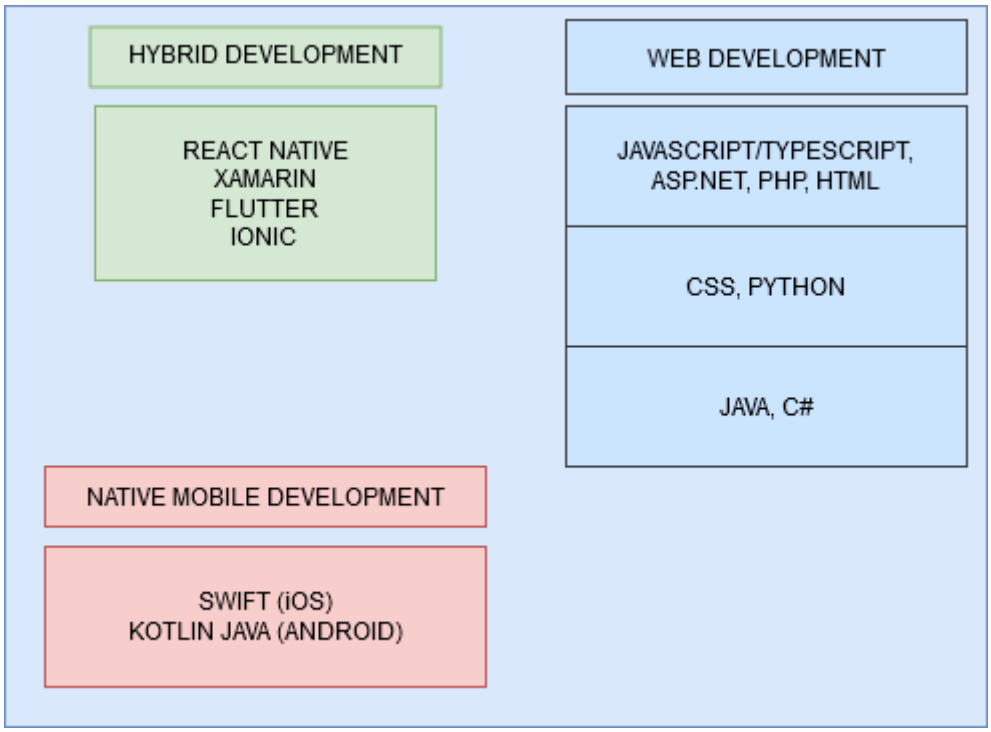

Figure 12 – Tech Options High-Level Diagram

Multiple technology options were considered including Native Mobile Development, Hybrid Development and Web Development. Native Mobile Development was rejected due to its inflexibility considering I wanted the system to work on laptops and tablets. Based on previous personal experience and a strong community, Web Development was preferred over Hybrid Development. This fulfils the requirement of the project being open-source, free and not vendor specific.

From the Web Development technologies, HTML, CSS, JavaScript, and TypeScript were selected.

3.2.2 Solution Architecture

The solution architecture of VisualPromptBuilder is predominantly focused on the client-side, ensuring swift and responsive interaction for users requiring assistive communication tools. The frontend is developed as a single-page application (SPA), where all user interactions occur seamlessly without needing page reloads.

The application does not currently utilise a database server. Instead, it leverages local storage capabilities, which house the pictogram data and ensure the system remains operational even in offline scenarios. This approach provides users with instantaneous access to the full range of pictograms and eliminates dependencies on internet connectivity, which can be crucial for users in environments with unstable internet access.

Despite the absence of a backend server, the application maintains a structured approach to managing data flow. A local TypeScript-structured data repository facilitates the organisation and retrieval of pictograms. The advantage is that the local data is performant and does not require external internet access. The use of web technologies such as the Web Speech API for speech synthesis and translation services via third-

party APIs demonstrates the system's capacity for integrating external services when available.

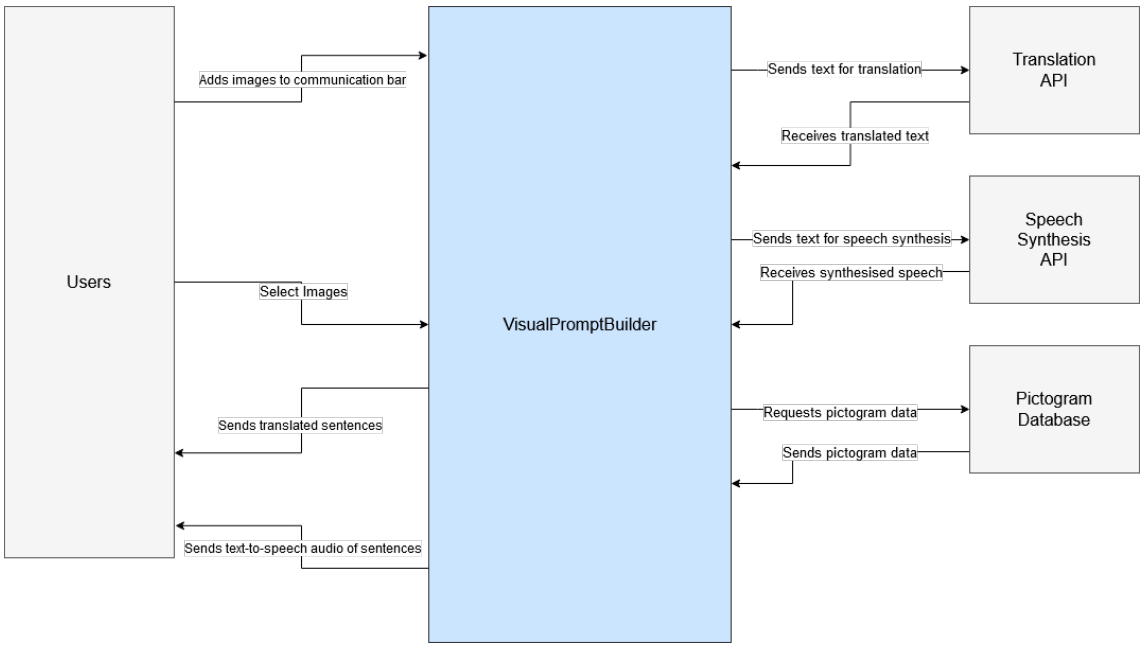

Figure 13 – Solution Architecture Context Diagram

3.3 System Features and Requirements

VisualPromptBuilder comes equipped with a suite of features tailored to aid nonverbal and limited-speech users. Integral to the design is the grid-based pictogram selection system. Users can browse and select from a variety of pictograms, which are categorised for ease of access and intuitive navigation. These pictograms can be combined within a communication bar to form coherent sentences.

Requirements for the system are established with the end-user in mind, emphasising accessibility and simplicity. The application is engineered to operate cross-platform, running effectively on desktop and mobile devices via web browsers, ensuring broad accessibility. To accommodate users from diverse linguistic backgrounds, the application integrates a translation feature, capable of converting sentences constructed from pictograms into multiple languages.

Moreover, the system is designed to output the formed sentences audibly using the integrated Web Speech API, which is crucial for users to communicate effectively with others. The requirement for real-time responsiveness is fundamental, with a robust local data management system, ensuring swift access to pictograms and immediate feedback from user input.

3.4 UI Design

Pernice et al. (2001) have shared the following design principles related to the user requirements and project:

3.4.1 Enhance Form Accessibility

To limit the amount of required information in forms, place text for field labels close to the fields, and use logical tab order. Considerations for colour use and avoiding reliance solely on colour to communicate information are also important.

3.4.2 Prioritise Text Clarity and Contrast

Choose text and background colours for good contrast. Ensure text can be magnified successfully and remains clear when viewed at larger sizes.

3.4.3 Optimise Page Organisation

Avoid causing user's screen to be cluttered with unnecessary information or navigational elements.

3.4.4 Ensure Content is Accessible

When graphics contain useful information, also provide that information in text.

The UI is crafted to offer an intuitive, engaging, and accessible experience. Adhering to Pernice et al. (2001) design principles, the UI presents a grid layout where pictograms are displayed in a visually coherent and organised manner. Each pictogram is accompanied by text labels to aid recognition and ensure the tool is usable for individuals with varying levels of literacy and cognitive ability.

Figure 14 – Pictogram Examples

Interaction with the UI is straightforward and forgiving. Users select pictograms to build sentences, and these selections populate a communication bar in the order chosen, with options to undo or delete selections as needed. The translation and speech synthesis features are seamlessly integrated into the UI, allowing users to convert their sentences into spoken words with a single click. The language selection dropdown is conveniently placed to switch between different languages effortlessly.

The design principles extend to ensuring the UI is responsive, adapting to different screen sizes and orientations to maintain usability across devices. Visual feedback is immediate, with animations and transitions that guide users through the sentence formation process without overwhelming them. Consistency in the use of icons, colours, and typography across the platform reduces the learning curve and enhances user retention.

3.5 Technology Stack

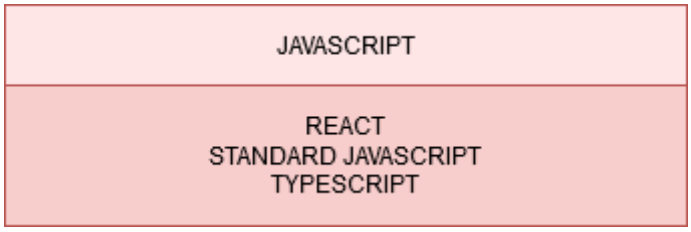

Figure 15 - Related JavaScript Technologies Used

Technology	Purpose	Reason for Choosing	Considered Alternatives
Next.js	Main application framework	Enables server-side rendering and static site generation, enhancing performance.	Gatsby.js: Next.js has a more comprehensive feature set out of the box.
React	UI library	Helps component-based architecture and efficient state management.	Vue.js: React's ecosystem and community support tipped the scales.
TypeScript	Programming language	Provides type safety and enhances development experience with autocompletion and error checking.	Plain JavaScript: TypeScript was preferred for its robust typing system.
Node.js	Runtime environment	Executes JavaScript code server-side, necessary for Next.js and build processes.	Deno: Node.js's widespread adoption and npm ecosystem made it the prime choice.
React Icons	Icon library	Supplies many icons from various packs, enriching the UI without bloating it.	Font Awesome: React Icons offered a wider variety and easier integration with React.
ESLint	Linting tool	Enforces code quality and consistency, crucial for maintainability and collaborative development.	JSLint: ESLint provides more configuration options and extensibility.
Cross-env	Scripting utility	Manages environment variables across different platforms, simplifying script configuration.	dotenv: Cross-env was selected for its simplicity in setting up scripts.

Table 5 – Technology Stack Overview

This stack was chosen for its ability to create an application that is high-performing, maintainable and scalable. Each piece was selected to enhance the development workflow; ESLint ensures code quality; TypeScript provides type safety. Next.js and React form the backbone of the application, chosen for their wide adoption and the rich ecosystem of tools and community support they bring. Node.js serves as the underlying platform to utilise these technologies, and React Icons adds visual clarity without sacrificing performance.

3.6 Web Technology Architecture

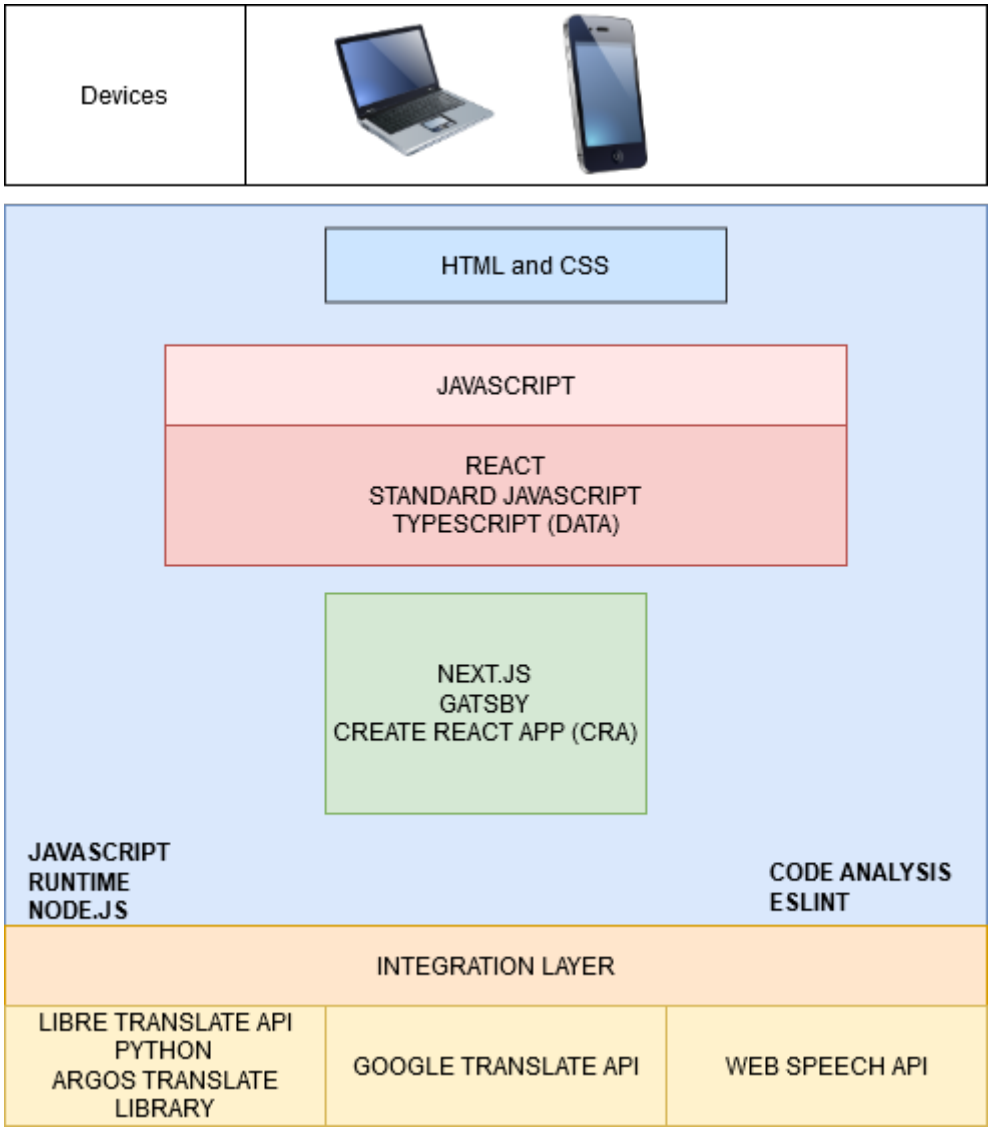

Figure 16 - Web Technology Architecture

When considering devices, the application should be a responsive and accessible web app, working on both laptops and tablets.

3.6.1 Benefits of Selected Web Application Technologies

3.6.1.1 TypeScript

TypeScript provided the following benefits to the project; it enhances error detection, has good IDE support, allows for efficient debugging, promotes robust, maintainable code, and integrates smoothly with modern frameworks.

3.6.1.2 React

React was chosen due to personal familiarity and ability to produce responsive websites and front ends. It is also free (open-source), future proof (security fixes are applied regularly), and has good community support - important when facing issues.

3.6.1.3 Next.js

Next.js enhances performance with optimised rendering by utilising caching. This would be useful for loading the pictograms. It streamlines development with built-in routing and image optimisation.

3.6.1.4 ESLint

ESLint provides the project with static code analysis, ensuring issues were detected early, and the code was error free. This was also useful when combined with TypeScript which was used to store the data.

3.7 Hardware Requirements

Component	Requirement
Operating System	Windows, macOS, Linux, or any OS, including tablet devices, supporting modern web browsers
Processor	No specific requirement, but a modern CPU is recommended for compatibility
Memory	At least 2 GB RAM, 4 GB or more recommended
Storage	Minimal; enough to run the operating system and web browser
Network	Internet connection for accessing web APIs and updates
Screen sizes	Laptop and Tablet: 1920×1080 recommended, however support scaling to other sizes.
Devices	Windows, Mac, iPad, Android tablets (large screen is ideal for nonverbal and cognitive limited users)

Table 6 - Hardware Specifications

3.8 Software Requirements

Development environment used Microsoft Visual Studio Code 1.88.1 on Windows 10.

Software	Version
Node.js	18.20.2
Next.js	14.2.1
React	18
React DOM	18
React Icons	5.1.0
Better-docs	2.7.3
Cross-env	7.0.3
ESLint	8
ESLint Config Next	14.2.1
JSDoc	4.0.2
TypeScript	5
Web Speech API	N/A
LibreTranslate API	1.5.6
Google Translate API	N/A

Table 7 - Software Specifications

3.9 Application Design including Considerations

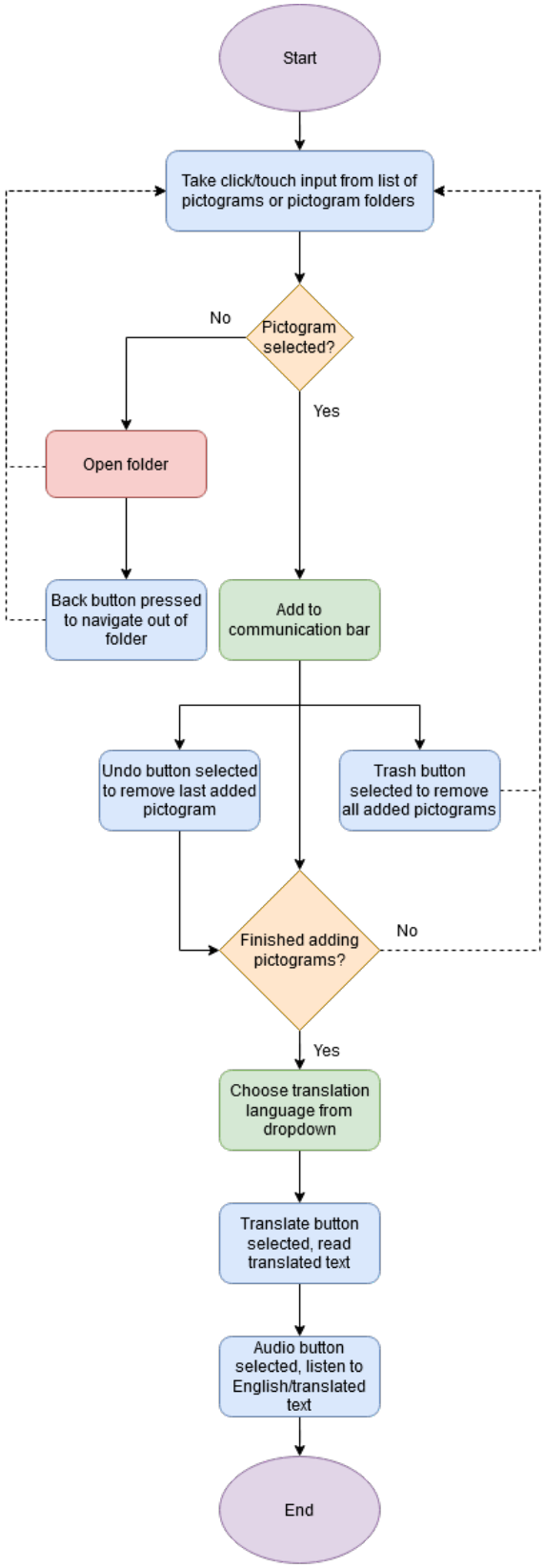

Figure 17 - Process Flow Diagram

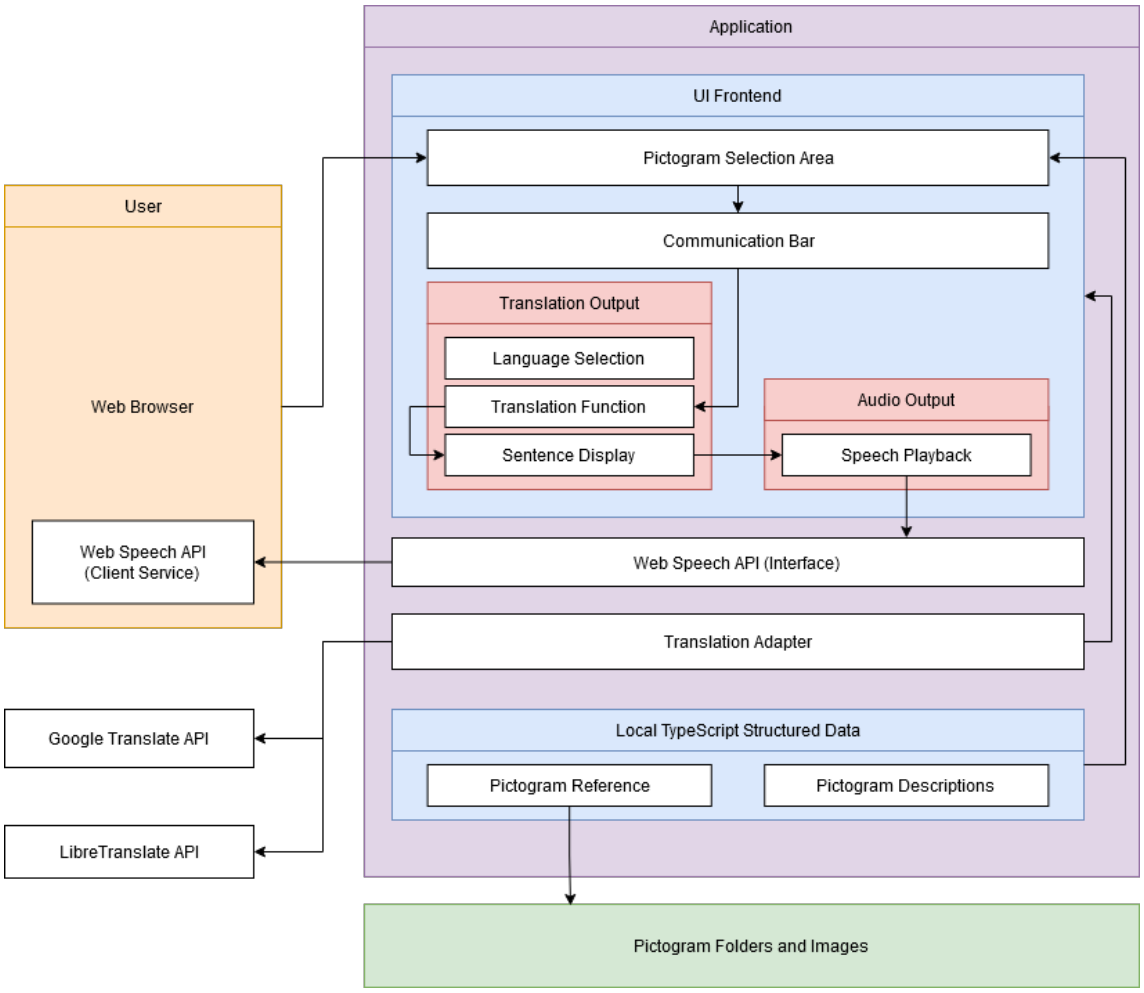

Figure 18 - Application Architecture Diagram

3.9.1 User Interface (UI)

3.9.1.1 Pictogram Selection Area

Using a web browser, the user selects pictograms (icons representing words or concepts).

3.9.1.2 Communication Bar

Shows options for output format selection (text, language selection, speech playback).

3.9.2 Application Services

3.9.2.1 Web Speech API (Interface-Client Service)

This interface in the application utilised browser-based API (implementation) converting text to speech for audio output.

3.9.2.2 Translation Function

Implementation of adapter to easily switch between translation service API, such as Google Translate or Libre Translate, with minimal code changes.

Takes pictogram's associated words and translates them into the desired language.

3.9.3 Data Processing

3.9.3.1 Local Data Storage (TypeScript)

TypeScript is used to store a data structure containing information about each pictogram location and the associated word.

Each object/entry would store:

- A file path pointing to the pictogram image stored locally.
- Word(s) associated with the pictogram, representing its meaning.

3.9.4 Output Channels

3.9.4.1 Translation Output

Displays the translated text on the screen for the user to read.

3.9.4.2 Speech Playback

Uses the Web Speech API to convert translated text into spoken audio.

3.9.4.3 Audio Output

Plays the spoken audio through the device's speakers or headphones.

Application Design Consideration

Design Patterns are good practice, based on proven application design approaches and industry experts. Over the years, there have been many design patterns utilised when architecting an application. The following section provides an overview of the design choice and reasoning.

Adapter and Facade Design Patterns

The facade pattern simplifies interactions with complex systems by providing a unified interface, thereby reducing subsystem complexity for client applications.

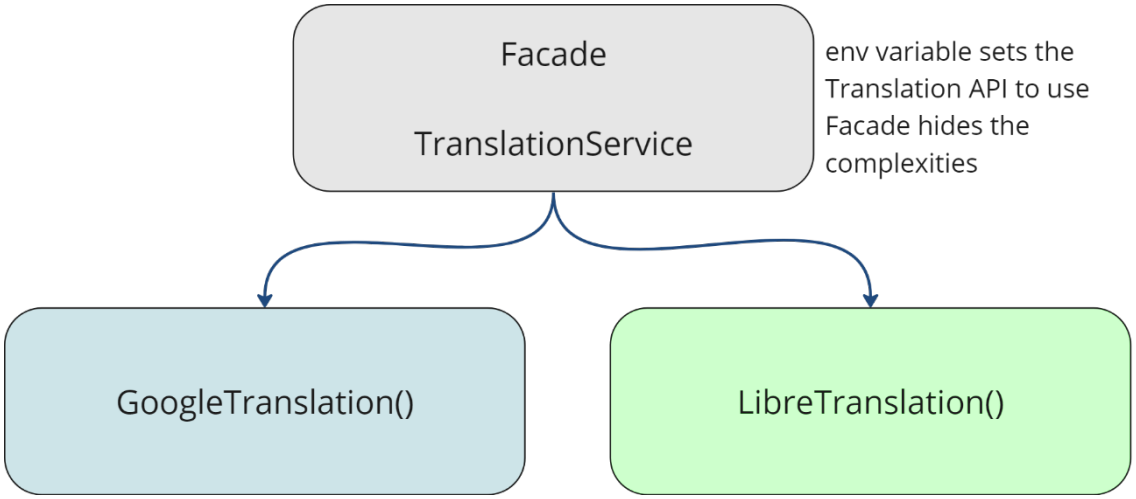

Figure 19 - Facade Pattern: Translation Service

Conversely, the adapter pattern resolves interface incompatibility between different parts of a program by converting one interface into another that clients expect.

Both patterns aim to streamline code management; however, the facade masks complexity, while the adapter rectifies compatibility issues, addressing distinct structural software design challenges.

3.9.5 Translation Adapter Pattern

VisualPromptBuilder used flexible and modular architecture - allowing for the integration of multiple translation services. The core of this functionality is facilitated by two primary adapters: Google Translation and Libre Translation. These adapters can be seamlessly switched within the application's environment, enabling the application to leverage different translation services without the need for a complete rebuild. This adaptability is crucial for the application's versatility in providing translation services.

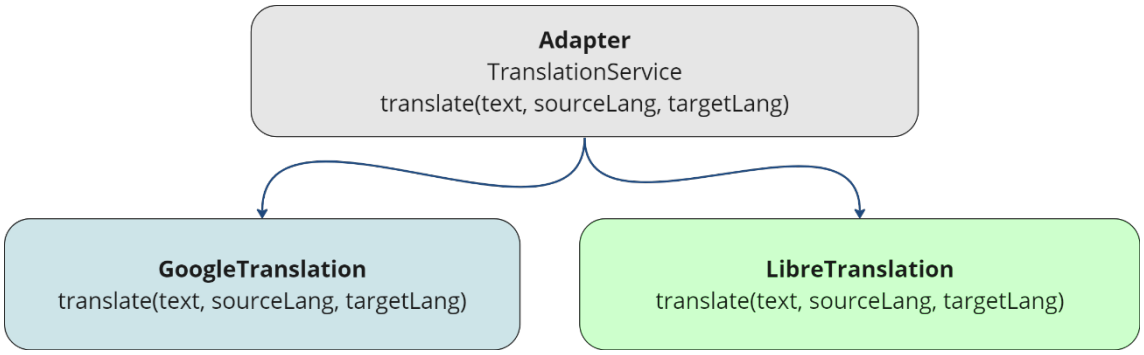

Figure 20 - Adapter Pattern: Translation Service

The switching of translation services is handled by the `TranslationService` class, which abstracts the complexity of interacting with different APIs behind a unified interface. (Refer to Appendix E)

The instantiation and use of the `TranslationService` within the application's components are demonstrated in Appendix F in TypeScript, showing how it integrates into the React component lifecycle to fetch and set the language options available to the user, and to perform the translation when a user requests it.

This configuration enables VisualPromptBuilder to maintain a robust and dynamic setup that is responsive to user needs and allows for the easy integration of future translation services, should the need arise. By abstracting the translation functionality, the application can be updated with minimal disruption, aligning with the principles of maintainability and scalability.

3.9.6 Text-to-Speech Integration

VisualPromptBuilder leverages the Web Speech API, a powerful browser-based tool that supports speech synthesis directly within the application. This integration is encapsulated within the `TextToSpeechService`, which abstracts the complexity of speech synthesis into a simple and easy-to-use service. This enhances the modularity of VisualPromptBuilder and aligns with modern web standards.

3.9.6.1 Using the Facade Pattern for Simplification

The `TextToSpeechService` acts as a facade, simplifying the use of the Web Speech API by providing a high-level interface for speech operations. This pattern allows other components of the application to perform speech synthesis without dealing with the underlying complexity of the Web Speech API's methods and properties. (Refer to Appendix G)

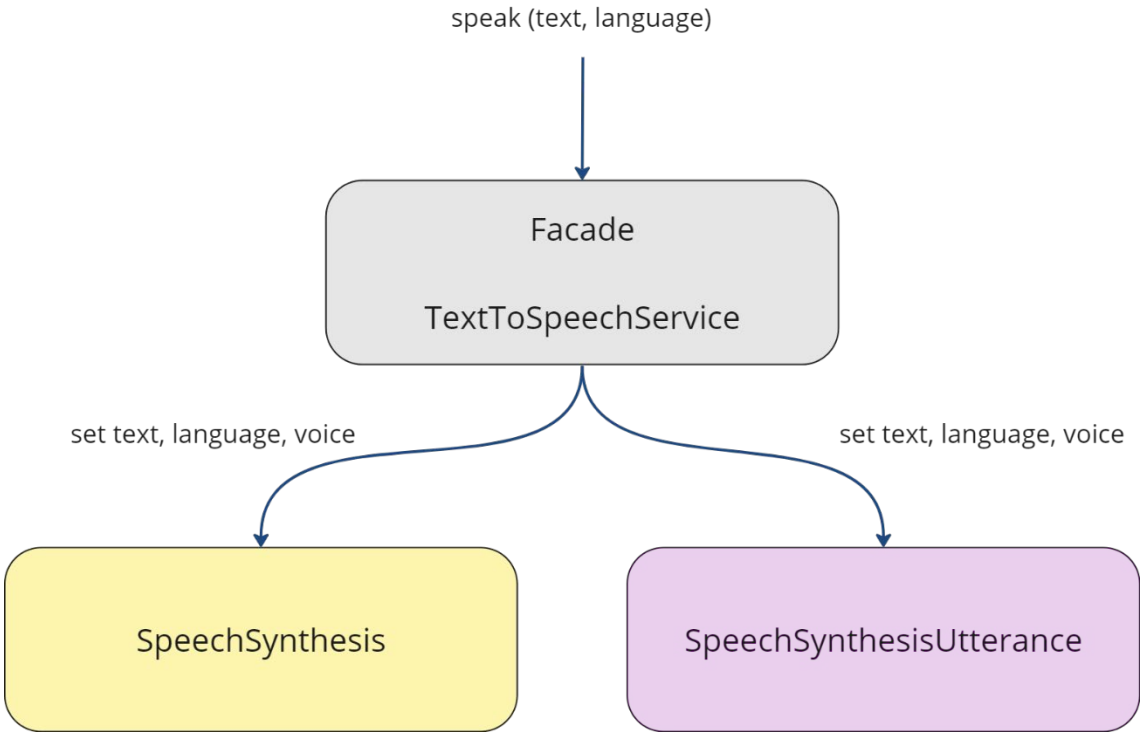

Figure 21 - Facade Pattern: Speech Synthesis

This class provides methods to set the language, voice, rate, and pitch of the speech, as well as to start and stop the speech. This encapsulation ensures that changes to the speech synthesis API or its implementation can be handled within this service without impacting other areas of the application.

3.9.7 Application of Adapter Pattern

While the `TextToSpeechService` primarily functions as a facade, it also incorporates the adapter pattern by allowing different properties of the `SpeechSynthesisUtterance` to be set in a standardised way, regardless of the browser's specific implementation. This makes the service more adaptable to different environments and extends its utility across various platforms.

3.9.8 Configuration Details

The `TextToSpeechService` is configured dynamically within the application's UI components, where it is instantiated and used as needed. This on-the-fly configuration allows for greater flexibility and responsiveness to user inputs, such as changing the language or adjusting the speech rate based on user preferences.

3.9.9 Example Usage in Application

Here's a TypeScript implementation of how the `TextToSpeechService` is utilised within a component to provide real-time text-to-speech functionality:

```
const handleTextTranslation = () => {
  const speaker = new TextToSpeechService();
  speaker.speak(translatedText, 'en-US');
};
```

This integration showcases the practical application of design patterns and modern JavaScript features to enhance the functionality and user experience of VisualPromptBuilder.

3.10 Application Walkthrough

VisualPromptBuilder was designed as a single-page web application, offering a seamless user experience without demanding navigation between different states or pages. The interface and features will be described, accompanied by relevant screenshots and diagrams. It supports both Click Mode for mouse and Touch Mode for interaction via touch screens.

3.10.1 Adaptive Design

The application automatically adapts to the user's browser settings, switching between light and dark modes based on the system's theme preferences. This smart feature eliminates the need for extra settings or buttons, whilst aligning with the user's established preferences.

3.10.2 Home Page and Pictogram Selection

Upon entering VisualPromptBuilder, users are greeted with a grid of pictograms, some categorised for easy identification and selection. Users can choose pictograms to build sentences that are displayed in a communication bar. The intuitive layout and responsive design accommodate both Click and Touch Modes, enhancing the application's accessibility.

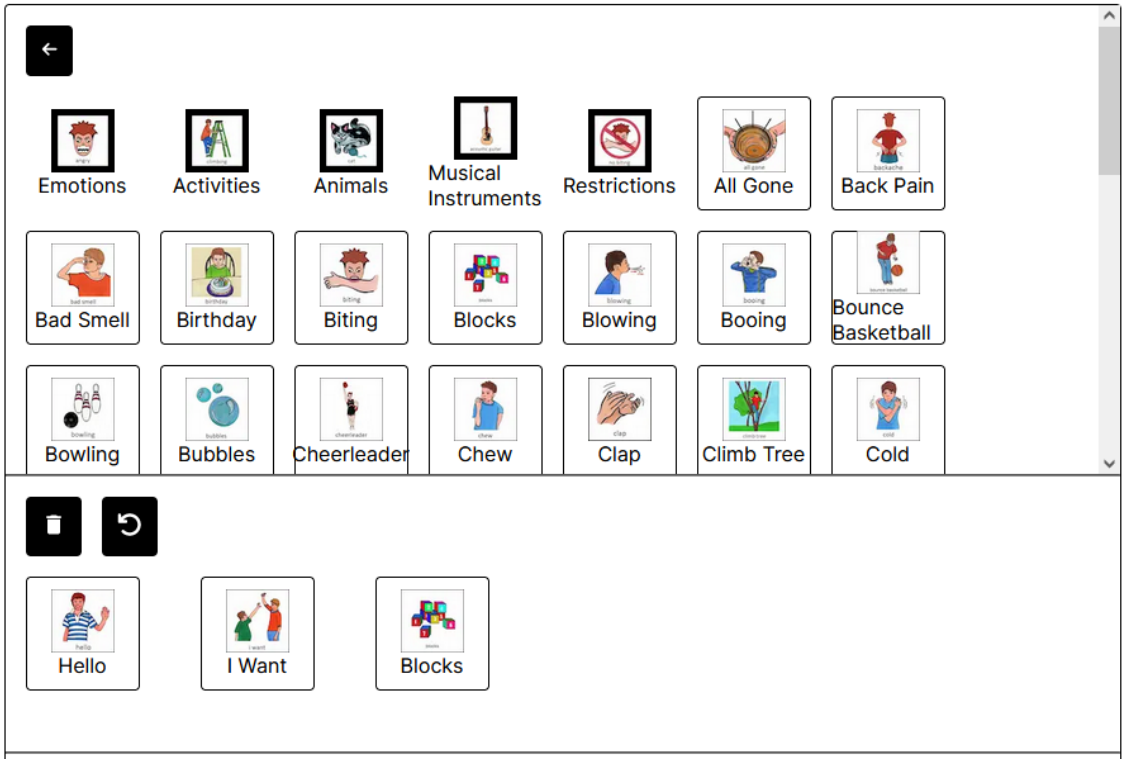

Figure 22 – Home Page & Pictogram Selection

3.10.3 Translation and Speech Synthesis

Below the communication bar, the application presents translation and speech synthesis options. Sentences formed from the selected pictograms can be translated into various languages. The system vocalises the translated sentence through the text-to-speech feature, enriching the communication experience by selecting the 'Audio' buttons near each associated sentence.

Figure 23 – Translation & Speech Synthesis

3.10.4 Responsive and Inclusive Experience

The application ensures an inclusive experience by providing dynamic, real-time visual feedback as sentences are constructed and translated. With automatic theme adjustment, VisualPromptBuilder aligns with the user's established visual preferences, offering a familiar and comfortable visual context, whether in light or dark mode. The Speech Synthesis API functionality allows users to have translated accents heard too, increasing inclusivity and relatability.

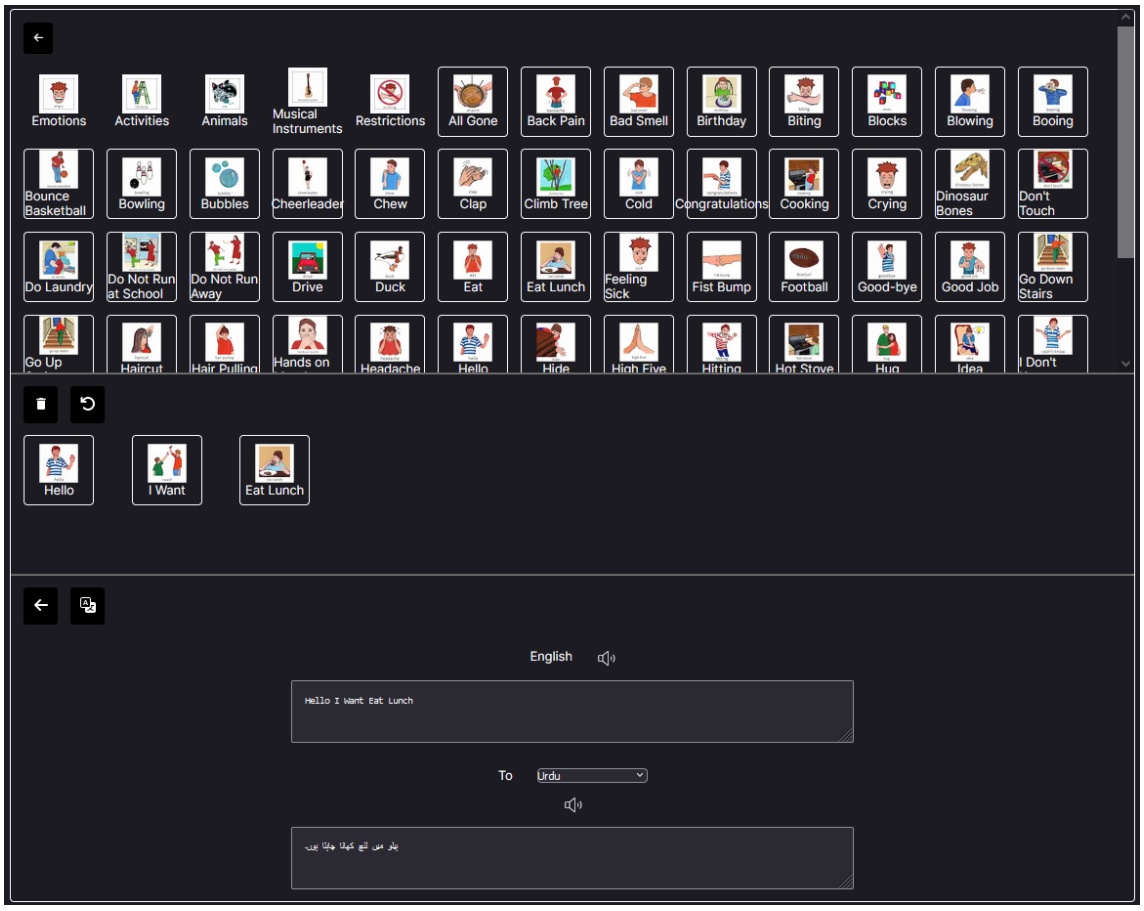

Figure 24 - Responsive Feedback (Dark Mode)

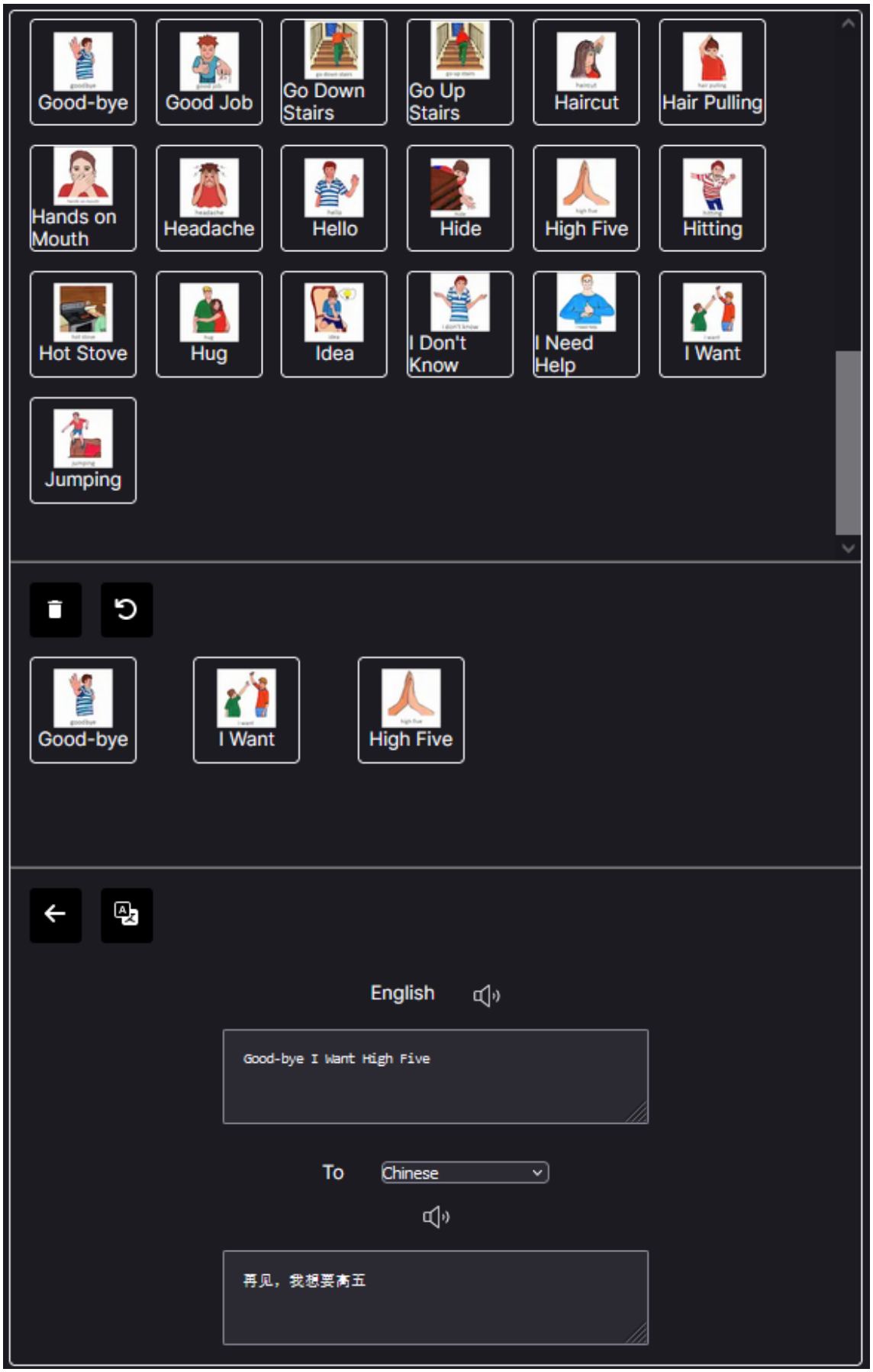

Figure 25 - Full Application View: User-Selected Theme

Chapter 4: **Development Process**

4.1 Software Development Lifecycle

When developing my project, a hybrid approach was employed, combining aspects of the Agile methodology with rigorous version control and build management. This approach was suited to the evolving nature of the project, which required flexibility in design and functionality enhancements based on ongoing feedback.

4.1.1 Methodology Explanation

Agile practices were integral in managing the development process and JIRA was used as the Agile project management tool. The project was broken down into three sprints, each aimed at delivering planned work. This iterative process allowed for continuous evaluation and adaptation of the project. Initial key requirements gathering, analysis and design was completed in the first sprint. However, even in the second sprint (development), as per agile, analysis and design improvements were incorporated. The final sprint was focused on testing, deployment, and documentation.

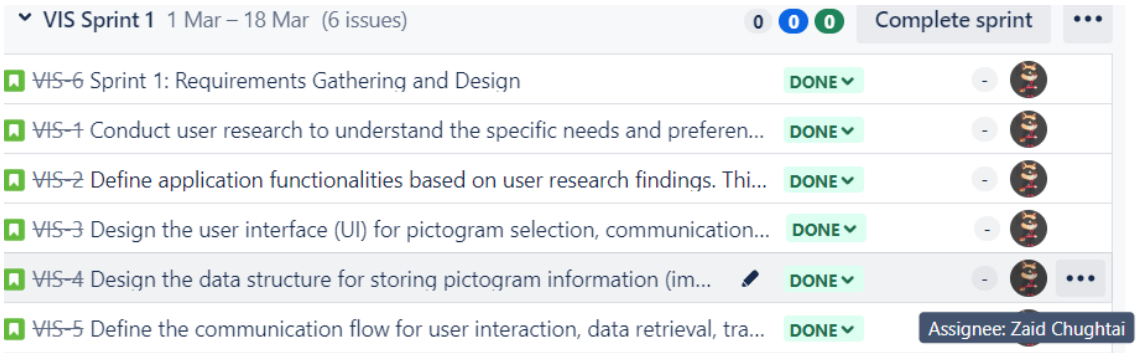

Figure 26 - Sprint 1: Agile Project Management

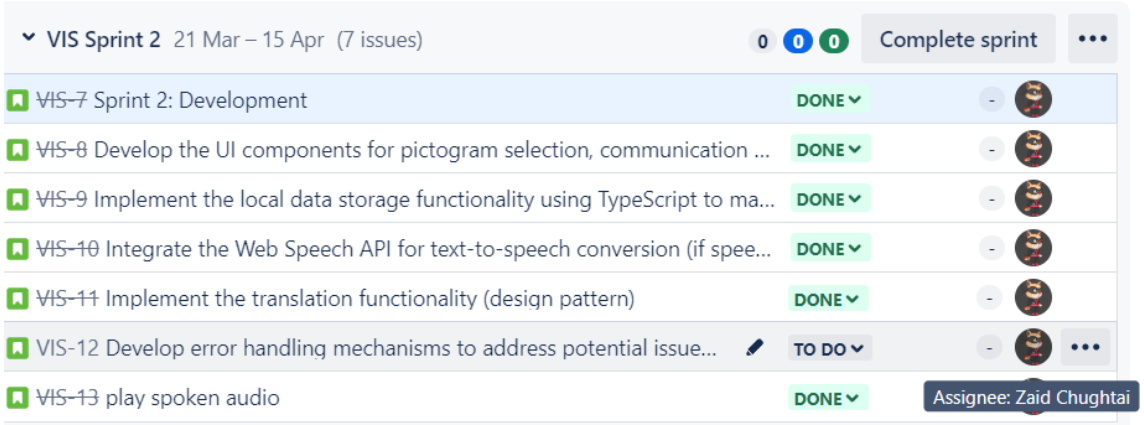

Figure 27 - Sprint 2: Agile Project Management

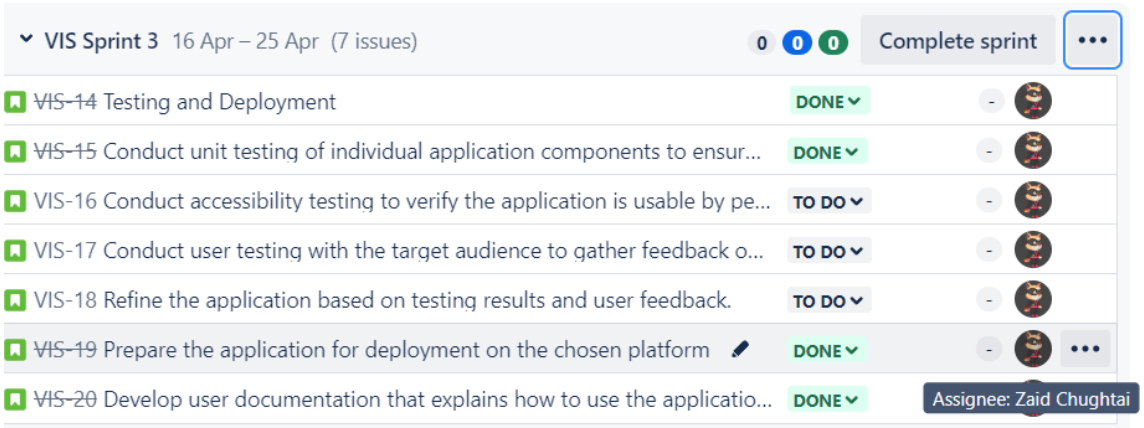

Figure 28 - Sprint 3: Agile Project Management

The Agile project management approach allowed minor changes to be included without having all the functional requirement details upfront.

4.2 Implementation

This section provides the deployment instructions, key operational features and specific configuration to run the solution.

4.2.1 Application Configuration

The application was designed to work with two different adapters for translation services—Google Translate and Libre Translate (an open-source alternative). This dual-adapter setup was managed through environment variables, which allowed for easy switching between services without modifying the core application code.

Here's how the configuration process was implemented:

4.2.1.1 Environment Setup

Environment variables were defined in a .env file to specify the active translation service.

```
# .env file
TRANSLATION_SERVICE=Google
```

4.2.1.2 Adapter Configuration

Based on the environment variable, the application loads the appropriate translation service adapter during the initialisation phase. (Refer to Appendix F)

4.2.2 Code Structure and Management

The application's codebase was structured to enhance maintainability and scalability.

Figure 29 – VSCode IDE Code Structure

This included:

4.2.2.1 TypeScript Integration

TypeScript was used throughout the project to improve code reliability and maintainability. Static type checking helped catch errors in the development process.

4.2.2.2 Design Patterns

Adapter and facade patterns were utilised to manage the translation services interchangeably, which simplified the integration of potentially more complex third-party APIs.

4.2.3 Translation Features

Translation and text-to-speech were critical features of the application.

4.2.3.1 Browser Compatibility

Efforts were made to ensure that text-to-speech functionality worked across different browsers, though variations in compatibility were noted.

4.2.4 Operational Details

Operational aspects of the application were carefully managed to support its frequent updating and deployment.

4.2.4.1 Build Exports

Each completed build was exported and archived. This not only provided backups of the application at various stages of its development but also allowed for quick rollbacks to previous versions when necessary.

4.2.5 Documentation and Support

To enhance the future maintenance of the project, documentation was generated. Tools like JavaScript Docs were used to automatically generate documentation, helping maintain a clear understanding of the codebase.

4.2.6 Future Proofing

Strategies for future-proofing the application included API extensibility; the architecture was designed to easily integrate new translation APIs, ensuring long-term scalability.

4.2.7 Deployment

The following instructions will allow the solution to be installed into the deployment environment. Currently the solution will be deployed on a local computer, outside the web browser using the Node.js JavaScript runtime application server. This architecture supports deployment to cloud and remote environments that support Node.js in the future.

Dependencies required

- Node.js Version 18.x
- (downloadable at https://nodejs.org/en/download or https://nodejs.org/dist/v18.20.2/node-v18.20.2-x64.msi)
- Follow the Node.js installer as it includes everything for a functional Node.js environment.
- Node.js will be Installed and by default use port 3000
- User also needs Web Browser (Microsoft Edge, Google Chrome 122.0 or later)

Installation

1. Download this project as a .zip and extract to your desired directory.
2. Navigate to the project directory using cd <project-directory>.
3. Install the necessary dependencies by running npm install --force.
4. Build the project by running npm run build:x. Where x is either google or libre depending on which translation API you would like to use.
5. Start the server by running npm run start:x. Where x is the chosen translation API.

Please replace <repository-url> and <project-directory> with the actual URL of the repository and the name of the directory respectively.

Run the solution
Go to the web browser address and type: http://localhost:3000

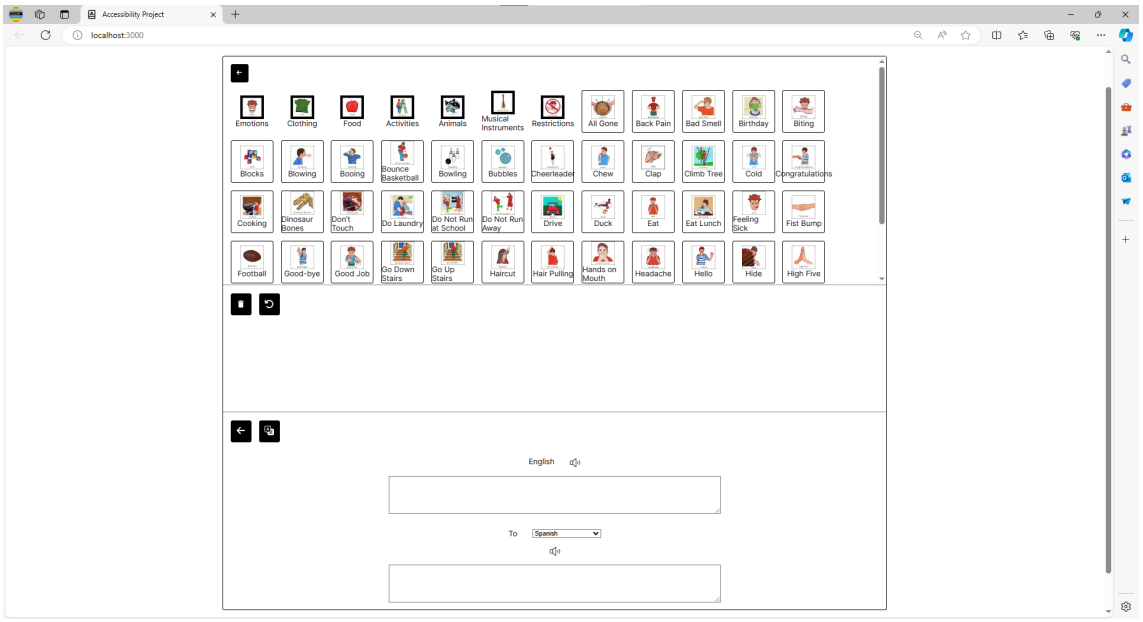

Figure 30 – Application running in Microsoft Edge

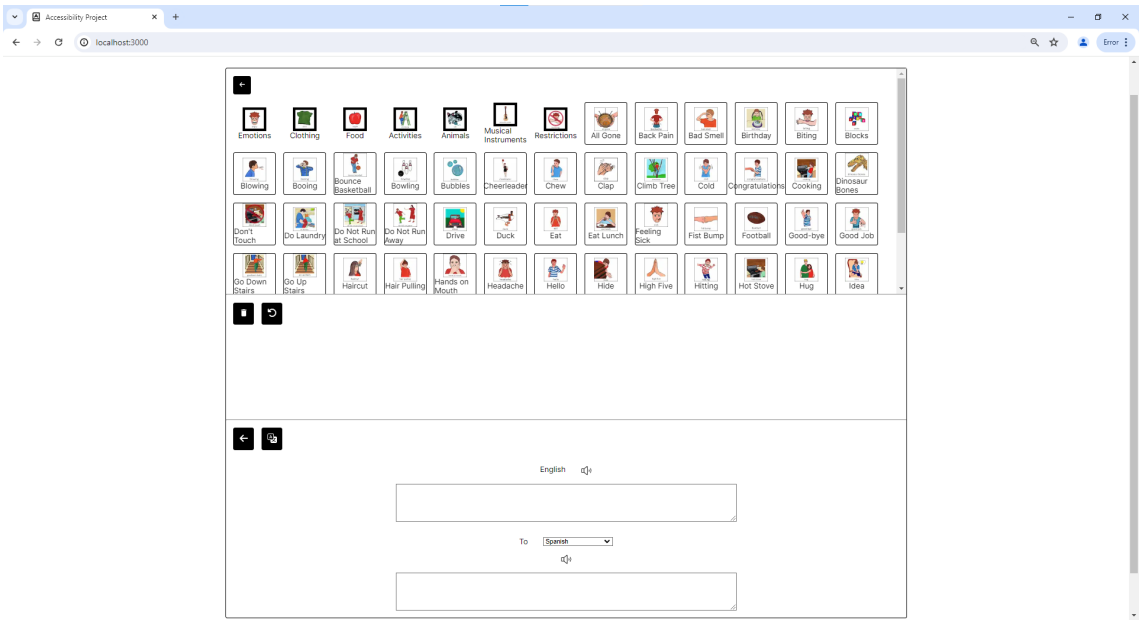

Figure 31– Application running in Google Chrome

Chapter 5: **Testing and Evaluation**

5.1 Testing Methodology

VisualPromptBuilder underwent a rigorous and systematic testing methodology to ensure its effectiveness, user-friendliness, and reliability. The initial phase of testing employed iterative testing, which allowed for quick feedback incorporation and continuous refinement of the application. As the primary tester, I conducted a thorough evaluation with each new iteration, assessing the application's functional performance, user interface design, and overall user experience.

A small group, consisting of 27 anonymous individuals, was selected to provide a service evaluation using Google Forms on the application's usability and functionality. This evaluation form was given to them right after they finished testing the application. The testers included a mix of genders, age groups, and technical backgrounds to ensure that the feedback was comprehensive and inclusive of different user experiences. The group tested each new version of the application and provided their insights, which were then used to make data-driven decisions for subsequent iterations.

The testing process for each iteration followed these steps:

5.1.1 Functionality Requirements Testing

The application's core features, such as pictogram selection, sentence formation, translation, and text-to-speech output, were evaluated for performance and accuracy.

Requirement ID	Functional Acceptance Criteria	Completed
1	Core Features Testing	Yes
2	Pictogram Selection, Sentence Building	Yes
3	Text Translation	Yes
4	Text-to-Speech Output	Yes
5	Output Options (display and audio, offline functionality)	Yes
6	Undo Last Added Pictogram	Yes
7	Clear Communication Bar/Generated Sentence	Yes
8	Error Handling (messages, crash recovery)	Yes

Table 8 - Functional Testing Results

5.1.2 Non-Functional Requirements Testing

Requirement ID	Description	Completed
1, 2	Performance (load time, response, absence of lag)	Yes

3, 4	Stability (no crashes, reliable navigation)	Yes

Table 9 - Non-Functional Testing Results

5.1.3 Usability Testing

Question ID	Question	Feedback Positive?
1	First impression of the application	Yes
2	Ease of understanding the application's purpose	Yes
3	Ease of finding features	Yes
4	Intuitiveness and logic of the layout	Yes
5	Learning curve for effective use	Yes
6	Confusion or difficulty with features	No
7	Overall satisfaction with usability	Yes
8	Likelihood of recommending the application	Yes
9	Ease of finding pictograms	Yes
10	Sensibility of the categorisation system	Yes
11	Ease of sentence building (clicking pictogram)	Yes

Table 10 - Usability Testing Outcomes

5.1.4 Accessibility Testing

Ensuring that the app was accessible to users with a range of abilities, including those with cognitive disabilities. Testing based on Web Content Accessibility Guidelines (WCAG) has been provided.

Result	Category	Requirement	Description
N	General	WCAG Conformance	Aim for WCAG 2.1 Level A & AA conformance for basic accessibility.
Y		Documentation	Provide clear, concise, and accessible user documentation
Y	Content	Text Alternatives	Ensure all non-text content (images, pictograms) has clear and concise alternative text descriptions.
Y		Colour Contrast	Verify sufficient colour contrast between text and background for low vision users.
N		Keyboard Accessibility	All functionalities should be accessible using a keyboard without relying solely on a mouse.
N		Focus Management	Keyboard focus should be clear and predictable when navigating the application.
N		Error Identification & Correction	Provide clear instructions and mechanisms for users to understand and correct errors.
Y	User Interface	Resize Text	Allow users to adjust text size to their preferences for better readability.
Y		Reflow	Ensure content reflows and resizes properly on different screen sizes and resolutions for optimal user experience (mobile, tablets, desktops).
Y	Additional Considerations	Screen Reader Compatibility	Test the application with screen readers to ensure accurate content presentation and smooth navigation.
N		User Testing	Conduct user testing with individuals with disabilities to identify and address any accessibility barriers.

Table 11 – WCAG Testing Checklist

5.2 Evaluation

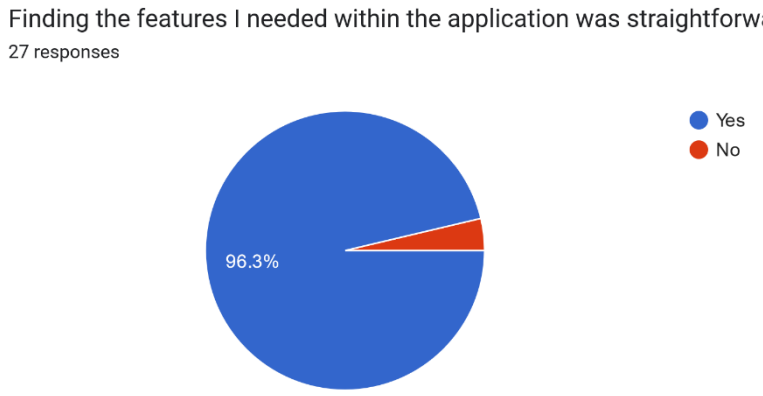

Figure 32 – Usability Service Evaluation: Feature Accessibility

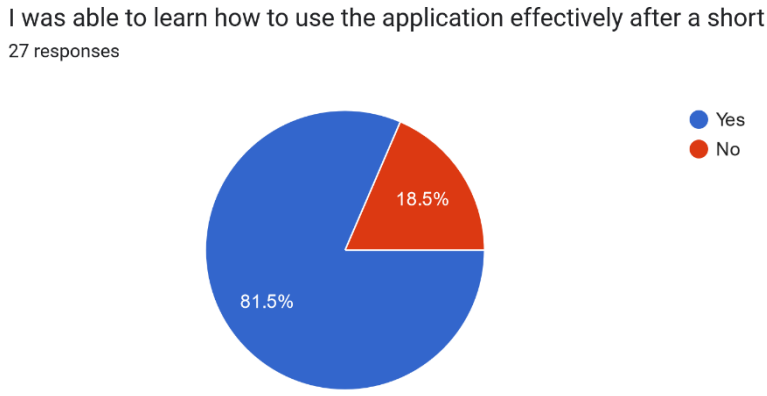

Figure 33 – Usability Service Evaluation: Learning Efficiency

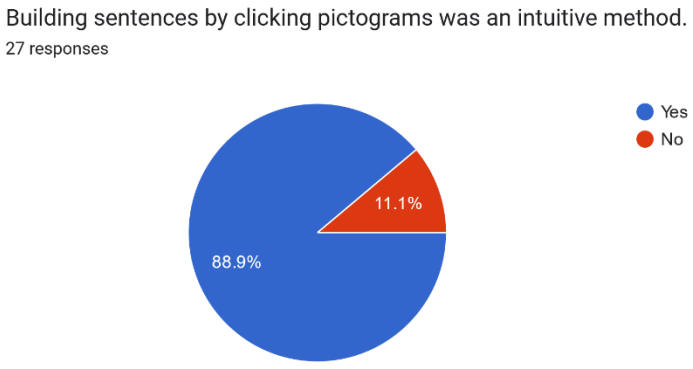

Figure 34 – Usability Service Evaluation: Sentence Building

From the findings of the usability test, 96.3% of the testers found the application straightforward and 81.5% were able to effectively use the application, after a short time.

88.9% of the testers found the building of sentences by clicking pictograms, as an intuitive method.

All the planned functional requirements were implemented and tested, as fully working.

Try/Catch was able to cope with some of the error testing; for future improvement, logging, and centralised error handling (with error codes) make finding and fixing issues easier. The were no performance or stability issues raised. From a usability and accessibility perspective, there were no major issues raised by the testers apart from not being able to solely use a keyboard.

Only basic security testing was conducted, as the solution was running locally via Node.js. For future releases, a security and penetration testing should be used that complies with OWASP requirements.

Conclusively, the testing results proved that the application was working as expected, by the different users that tested it. As a prototype, future releases should be tested with nonverbal users and individuals with cognitive issues.

Chapter 6: **Discussion**

6.1 Functional Objectives

From a functional objectives' perspective, research was conducted, as was an interview with an expert.

Following objectives were completed successfully:

- Text-to-speech spoke sentences.
- Responsive design ensured smooth operation across platforms (mobile, web).
- The application offered multilingual support through translation and speech output.
- Modular architecture with well-documented code enabling future extension.

6.2 Related to the Research Questions

How can pictogram-based communication tools be optimised for ease of use and effectiveness for nonverbal users?

- Clear, consistent pictograms with logical organisation for easy browsing.
- Simple selection and sentence building tools.
- Multiple representation options (text labels, pictogram folders).

What are the technical requirements for integrating text-to-speech technology that accurately and naturally vocalises sentences formed from pictograms?

- Accurate translation of pictograms to natural language text.
- High-quality speech synthesis with natural intonation and pronunciation (multilingual).
- User customisation of speech rate, volume, and voice type – currently requires paid services.
- Offline functionality with downloadable speech database - currently requires paid services.
- Utilisation of established text-to-speech APIs for efficiency.

In what ways can the user interface be designed to accommodate users with varying abilities and preferences, enhancing accessibility and customisation?

- Web Content Accessibility Guidelines (WCAG) compliance for screen reader compatibility and assistive technology support.
- Support for touch screens, keyboards, and alternative input devices.

- Clear visual hierarchy and high contrast for improved readability.
- Multi-lingual support with user-friendly interface language switching.

What are the challenges associated with developing a cross-platform communication tool, and how can these be effectively addressed?

- Device compatibility across tablets, and computers (Android, iOS, Mac, Windows).
- Use of cross-platform development frameworks for code reusability.
- Responsive layout design for optimal experience across different screen sizes and resolutions.
- Performance optimisation for smooth operation on devices with varying processing power and memory.

6.3 Limitations of the research

The aim and objectives were accomplished; but the project was more complex than previously envisioned. It would have been possible to spend an entire project researching the communication tools that are available for nonverbal communication and cognitively disabled individuals. As technologies are continuously advancing, research could have focused on a comparative study of new web development technologies and frameworks. By combining research elements, solution comparison (of readily available solutions), feature comparison and then the development of a new competing solution, with new features like translation and speech was a daunting task. Further rigorous research could have been completed in the different aspects mentioned. The Design Thinking approach encourages user involvement and discussion, however one challenge faced is how to deal with situations when the user cannot communicate.

With hindsight, multiple interviews and discussions could have been arranged with domain experts, however, this was not possible because of time constraints. Technically, there was a steep learning curve of new technologies, techniques, and libraries, despite JavaScript development already being known.

Chapter 7: **Conclusion and Future Work**

7.1 Future research and work

7.1.1 Design Thinking and Empathy Maps

As part of the Design Thinking approach, one key aspect is to create multiple Personas, Empathy maps and Customer Journey Maps. This is a challenge though, as nonverbal and cognitive limited people will not be able to express themselves in an interview. Observations should be made of the potential users utilising different related solutions, to capture aspects that cause issues. By asking the authorised caregiver for permission, usage metric data can be captured in a future prototype iteration. This would allow for tracking of the time taken to click different buttons or if certain functionality is not even utilised. Another approach, subject to having permissions granted, is to utilise eye tracking, to see which aspects of the solution are engaging and are seen. This will also allow reporting on attention span and potential user experience issues, that are not verbalised.

7.1.2 Security and Load Testing

More detailed security and penetration testing should be incorporated in the future, to ensure that the application stays compliant with web application standards such as Open Web Application Security Project (OWASP, 2021). This is important, as in the future, a login can be implemented, which allows preferences and frequently used symbols/sentences to the accessed.

Load testing should be incorporated into the future, to make sure that the application performs well under different load.

7.1.3 Data Storage

From an application architecture perspective, TypeScript local data should be replaced with either external JSON or database. This would allow functionality to dynamically add pictograms and words. The other advantage of using an external database, would be allow multiple concurrent users, letting users utilise the services and letting caregivers receive reports and share suggestions.

7.1.4 AI Implementation

With the rise of Generative AI, another future improvement could be to utilise Image-to-Text AI conversion or even dynamic generation of symbols. Currently, there are no free services that support the dynamic generation of symbols based on corresponding words - this could be an area of future research. Web Speech API has an implementation directly on the web browser of the user, whilst convenient as no additional service is required, it currently sounds robotic. As speech synthesis improves, either this will be updated, or another better service can be incorporated. It will be good to utilise a Generative AI service to generate a full sentence (with words like "the", allowing "I want to play with the blocks"), as Natural Language Process and Generation have advanced.

7.1.5 Further research

Additional future work could incorporate the caregiver in agile daily stand-ups and sprint reviews (as part of the software development), however, due to not being able to access the users, this was currently not possible. While qualitative research included an interview with one expert, future plans should aim for 10 interviews and 30 observations

of nonverbal users. Despite this expanded scope, the findings still may not be generalisable due to time constraints.

7.2 Conclusion

VisualPromptBuilder was developed intending to mark an advancement in assistive technology for nonverbal people. Throughout this project, I concentrated on creating a tool that both addresses the communication barriers faced by these individuals and enhances their ability to interact with the world around them meaningfully. The project faced many challenges and learning opportunities and I know I have developed in my thinking along the way.

My approach, grounded in Design Thinking, accentuated empathy and user centric design, ensuring the tool was not just functional but also intuitive and accessible for its users, considering their diverse needs. The approach, empathy map, user journey map and the research work made me appreciate the challenges and pain points that carers and nonverbal individuals face daily.

The technology selection was correct to choose JavaScript modern web technologies that provided responsive interfaces in different sized devices.

I believe that the research conducted, and the developed solution (VisualPromptBuilder) assists in bridging communication gaps and provides a good foundation for further research and work. It was tempting to implement more AI functionality, however, as my research and analysis continued, I found this to be ethically and responsibly a lower priority than addressing needs of the nonverbal and disadvantaged community of users. It embodies a step forward in making everyday interactions more accessible for nonverbal users, with an intention of contributing to their autonomy and social inclusion. I have really enjoyed this project and I look forward to future challenges with an open mind.

Chapter 8: **References**

Barua, P.D., Vicnesh, J., Gururajan, R., Oh, S.L., Palmer, E., Azizan, M.M., Kadri, N.A. and Acharya, U.R., 2022. Artificial intelligence enabled personalised assistive tools to enhance education of children with neurodevelopmental disorders—a review. *International Journal of Environmental Research and Public Health*, *19*(3), p.1192.

https://www.mdpi.com/1660-4601/19/3/1192

Carlgren, L., Rauth, I. and Elmquist, M., 2016. Framing design thinking: The concept in idea and enactment. *Creativity and innovation management*, *25*(1), pp.38-57.

https://www.academia.edu/download/107918511/Carlgren_20Rauth_20Elmquist_202016_20CIM_20Framing_20Design_20Thinking_20.pdf

Dyzel, V., Oosterom-Calo, R., Worm, M. and Sterkenburg, P.S., 2020, September. Assistive technology to promote communication and social interaction for people with deafblindness: a systematic review. In *Frontiers in Education* (Vol. 5, p. 578389). Frontiers Media SA.

https://www.frontiersin.org/articles/10.3389/feduc.2020.578389/full

Li, X. and Zhao, X., 2015, December. The Application of Nonverbal Symbols in Intercultural Business Communication. In 2015 International Conference on Social Science and Higher Education (pp. 198-201). Atlantis Press.

https://www.atlantis-press.com/article/25843332.pdf

Lohmann, M.J., Hovey, K.A., Gauvreau, A.N. and Higgins, J.P., 2019. Using assistive technology tools to support learning in the inclusive preschool classroom. *The Journal of Special Education Apprenticeship*, *8*(2), p.5.

https://scholarworks.lib.csusb.edu/josea/vol8/iss2/5/

Morris, M.A., Meier, S.K., Griffin, J.M., Branda, M.E. and Phelan, S.M., 2016. Prevalence and etiologies of adult communication disabilities in the United States: Results from the 2012 National Health Interview Survey. Disability and health journal, 9(1), pp.140-144.

https://www.sciencedirect.com/science/article/abs/pii/S1936657415001016

Motti, L., Vigouroux, N., & Gorce, P., 2015. Improving Accessibility of Tactile Interaction for Older Users: Lowering Accuracy Requirements to Support Drag-and-Drop Interaction. , pp. 366-375.

https://doi.org/10.1016/j.procs.2015.09.281.

Okamoto, M., Kojima, R., Ueda, A., Suzuki, M. and Okuno, Y., 2022. Characterizing eye-gaze positions of people with severe motor dysfunction: Novel scoring metrics using eye-tracking and video analysis. *Plos one*, *17*(8), p.e0265623.

https://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0265623&type=printable

OWASP (2021) OWASP Top Ten Web Application Security Risks. Available at: https://owasp.org/www-project-top-ten/

Pernice, K., Nielsen, J., Farrell, S., Mizobuchi, S., Ishida, N., Stover, U.A., Yohay, M., Franko, E. and Richardson, A., 2001. Usability guidelines for accessible web design. Evidence–Based User Experience Research, Training, Consulting.

https://media.nngroup.com/media/reports/free/Usability_Guidelines_for_Accessible_Web_Design.pdf

Peters, D., Vold, K., Robinson, D. and Calvo, R.A., 2020. Responsible AI—two frameworks for ethical design practice. IEEE Transactions on Technology and Society, 1(1), pp.34-47.

https://ieeexplore.ieee.org/iel7/8566059/8995808/09001063.pdf

Plomin, R., Price, T.S., Eley, T.C., Dale, P.S. and Stevenson, J., 2002. Associations between behaviour problems and verbal and nonverbal cognitive abilities and disabilities in early childhood. Journal of Child Psychology and Psychiatry, 43(5), pp.619-633.

https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=e41efffedac86fe41e479ac0bfd34dbed5d8b24d

Raharjo, F., Ridwan, R., Harianto, D., Jailani, S., Ulfah, S. and Sari, F., 2022, February. Nonverbal Communication Behavior of Autistic Children in the Therapy Process at the Center Jambi Province Autism Service. In Proceedings of the 4th International Colloquium on Interdisciplinary Islamic Studies in conjunction with the 1st International Conference on Education, Science, Technology, Indonesian and Islamic Studies, ICIIS and ICESTIIS 2021, 20-21 October 2021, Jambi, Indonesia.

https://eudl.eu/pdf/10.4108/eai.20-10-2021.2316436

Scherer, M.J., Hart, T., Kirsch, N. and Schulthesis, M., 2005. Assistive technologies for cognitive disabilities. *Critical Reviews™ in Physical and Rehabilitation Medicine*, *17*(3).

https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=63efc03b888ee1fb564a810cf3c4c3e6c79451ce

SHAFEE, R.A., HUDA, R., HOSSAIN, M.I., SHOHAG, M.M.H. and KHAN, S.I., 2022. Improving the Treatment Process of Bengali Autistic Children using Specialized Mobile Application. *Database Systems Journal*, *13*.

https://dbjournal.ro/archive/33/33.pdf#page=30

Shinohara, K., Bennett, C.L. and Wobbrock, J.O., 2016, October. How designing for people with and without disabilities shapes student design thinking. In Proceedings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility (pp. 229-237).

https://dl.acm.org/doi/pdf/10.1145/2982142.2982158

Zashchirinskaia, O., 2020. Nonverbal Communication as a Means of Social Integration: The Development of Nonverbal Communication in Primary Schoolers with Intellectual Disabilities. *Journal of Intellectual Disability—Diagnosis and Treatment*, *8*(4), pp.610-618.

https://pdfs.semanticscholar.org/af0b/11601ed27445f9ac7ad1a816d3e12ef96483.pdf

Applications Accessed

ChatGPT: https://chat.openai.com/ (Accessed: [13/12/2023])

Bard (now Gemini): https://bard.google.com/ (Accessed: [13/12/2023])

Widgit Software: https://www.widgit.com/index.htm (Accessed: [12/03/2024])

Widgit Software - InPrint 3: https://www.widgit.com/products/inprint/index.htm (Accessed: [12/03/2024])

Boardmaker: https://goboardmaker.com/ (Accessed [12/03/2024])

My PECS (2024) *Free PECS Cards*. Available at: http://www.MyPECS.com (Accessed: [14/03/2024])

Pictalk AAC: Talk with pictograms for autism and more. Available at: https://www.pictalk.org/ (Accessed: [24/03/2024])

Pictalk Open-Source Repository: https://github.com/Pictalk-speech-made-easy/pictalk-frontend (Accessed: [24/03/2024])

Appendices

Appendix A – Terminology Glossary

Term	Description
VisualPromptBuilder	A communication tool developed for nonverbal individuals, using pictograms, text-to-speech, and translation technologies.
Nonverbal communication	Communication without spoken language, often through gestures, body language, or visual aids like pictograms.
Cognitive disabilities	Disabilities that affect mental processes such as learning, reasoning, and memory.
Pictograms	Symbols or images that represent an object, action, or concept, used to aid communication.
Text-to-speech (TTS)	Technology that converts written text into spoken words, used to aid individuals with reading or speaking difficulties.
Translation technologies	Tools and software that convert text or speech from one language to another.
User interface (UI)	The visual and interactive components of a software that allow users to interact with the application or device.
User experience (UX)	The overall experience a user has with a product or service, especially in terms of how easy and pleasant it is to use.
Web Speech API	A browser API that enables voice data to be incorporated into web apps for voice recognition and speech synthesis.
Assistive technology	Devices or systems that help maintain or improve an individual's functioning and independence, enhancing quality of life.
System architecture	The conceptual model that defines the structure, behaviour, and more views of a system.

Single-page application (SPA)	A web application that interacts with the user by dynamically rewriting the current page rather than loading entire new pages.
Local storage	Web storage that allows sites to store data in a web browser on the device used by the user.
TypeScript	A programming language developed and maintained by Microsoft, a typed superset of JavaScript that compiles to plain JavaScript.
Agile methodology	A method of project management that involves a continuous iteration of development and testing in the software development lifecycle.
Facade pattern	A structural design pattern that provides a simplified interface to a complex system, making the system easier to use for clients.
Adapter pattern	A software design pattern that allows objects with incompatible interfaces to collaborate.
React Context	A component structure provided in the React framework to share values like preferences and themes across the component tree.
Web browsers	Software applications used to access information on the World Wide Web.
Multilingual support	The capability of software to allow the use of multiple languages for input and output.
Speech synthesis	The artificial production of human speech, often used in conjunction with text-to-speech technologies.
API (Application Programming Interface)	A set of rules and tools for building software applications, specifying how software components should interact.
Next.js	An open-source development framework built on top of Node.js enabling React based web applications functionalities.
React	A JavaScript library for building user interfaces, maintained by Facebook and a community of individual developers and companies.

Node.js	An open-source, cross-platform, JavaScript runtime environment that executes JavaScript code outside a web browser.
Software development lifecycle (SDLC)	The process of planning, creating, testing, and deploying an information system.
Frontend	The part of a website or software that users interact with directly.
Backend	The part of a website or software that is not directly accessed by the user, typically responsible for storing and managing data.
Server-side rendering	The process of rendering a web page on the server instead of in the browser, which can improve performance and SEO.
Static site generation	The process of compiling and rendering a website or application at build time, typically improving load times and SEO.
Component-based architecture	An architectural style of software engineering that divides functionalities into smaller, reusable components.
State management	In software engineering, refers to managing the state (i.e., the data) of one or more user interface controls like text fields, etc.
Linting	The process of running a program that will analyse code for potential errors.
Cross-platform compatibility	The capability of software to run identically on different computing platforms.
Scalability	The capability of a system to handle a growing amount of work by adding resources to the system.
Maintainability	The ease with which a software can be maintained in order to correct defects, improve performance, or adapt to a changed environment.
Version control	A system that records changes to a file or set of files over time so that specific versions can be recalled later.

Deployment	The process of making a software application available for use.
Code quality	A set of characteristics that determine the ability of the software to meet its requirements and user expectations.
Configuration	The arrangement of components or elements in a particular form, figure, or combination.
Environment variables	Variables that are configured outside the program, typically used to determine behaviour settings like database connections.
Accessibility	The design of products, devices, services, or environments to be usable by people with disabilities.
Responsive design	An approach to web design aimed at crafting sites to provide an optimal viewing experience across a wide range of devices.
Inclusive design	Designing products or environments to be usable by all people, to the greatest extent possible, without the need for adaptation.
Iterative testing	A method of software testing where progressively smaller parts of the software are tested to ensure functionality.
User testing	The process in which real users test a product to identify design flaws and usability issues before it is launched.
Feedback loop	A system where outputs of a system are circled back and used as inputs.
Performance evaluation	The process of assessing the performance of a software application or system against specific criteria.
PECS (Picture Exchange Communication System)	A form of augmentative and alternative communication typically used by individuals with autism spectrum disorders.
ChatGPT	AI model developed by OpenAI that interacts in a conversational manner.
Bard	AI model developed by Google. As of February 2024, Bard has been updated and renamed to Gemini.

Large Language Models (LLMs)	A large language model is notable for its ability to achieve general-purpose language generation. Examples used in this report are ChatGPT and Bard.
Prompt engineering	The practice of designing and refining prompts to efficiently generate desired responses or actions from an AI system.

Appendix B – First iteration of application

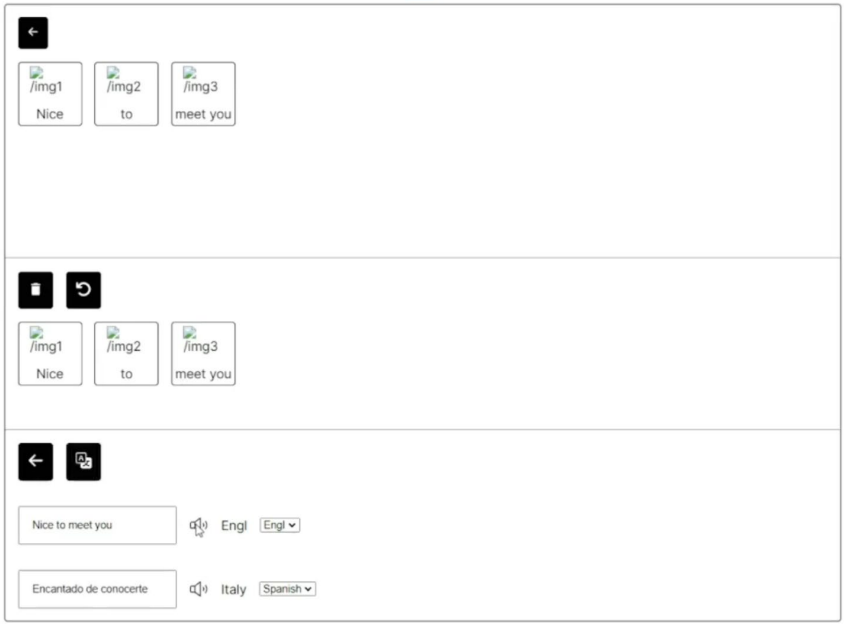

Figure 35 – Initial Prototype Wireframe

Appendix C – Testing Dataset Creation using GPT-4 Image Recognition

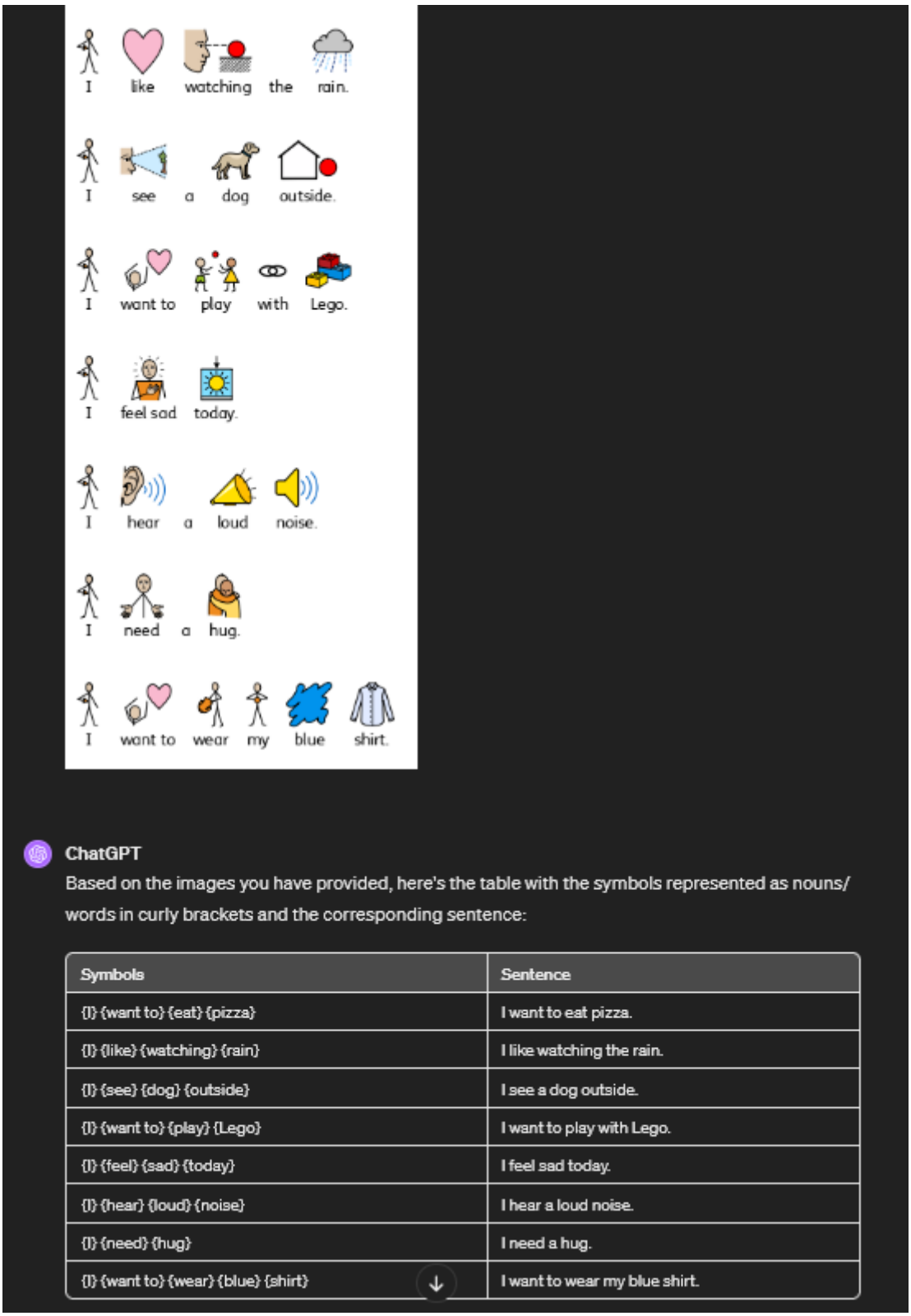

Figure 36 – ChatGPT-4 Image Recognition Test: Thread to Sentence

Appendix D – Generation of sentences using threads and cues in LLMs

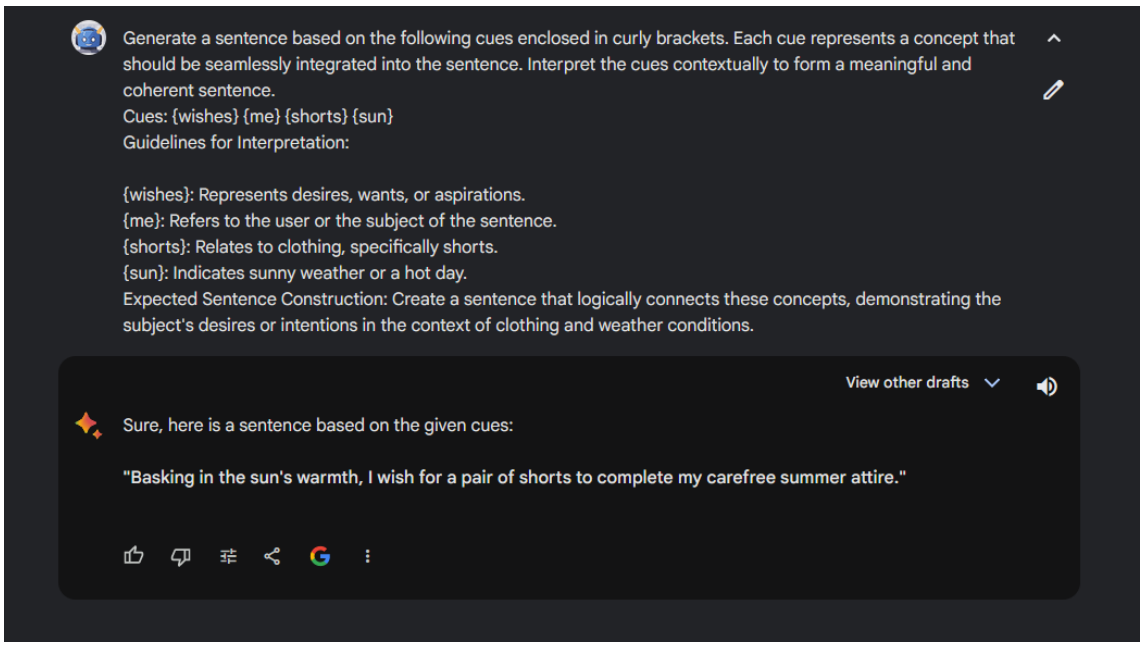

Figure 37 – Bard: 1st Sentence Generation

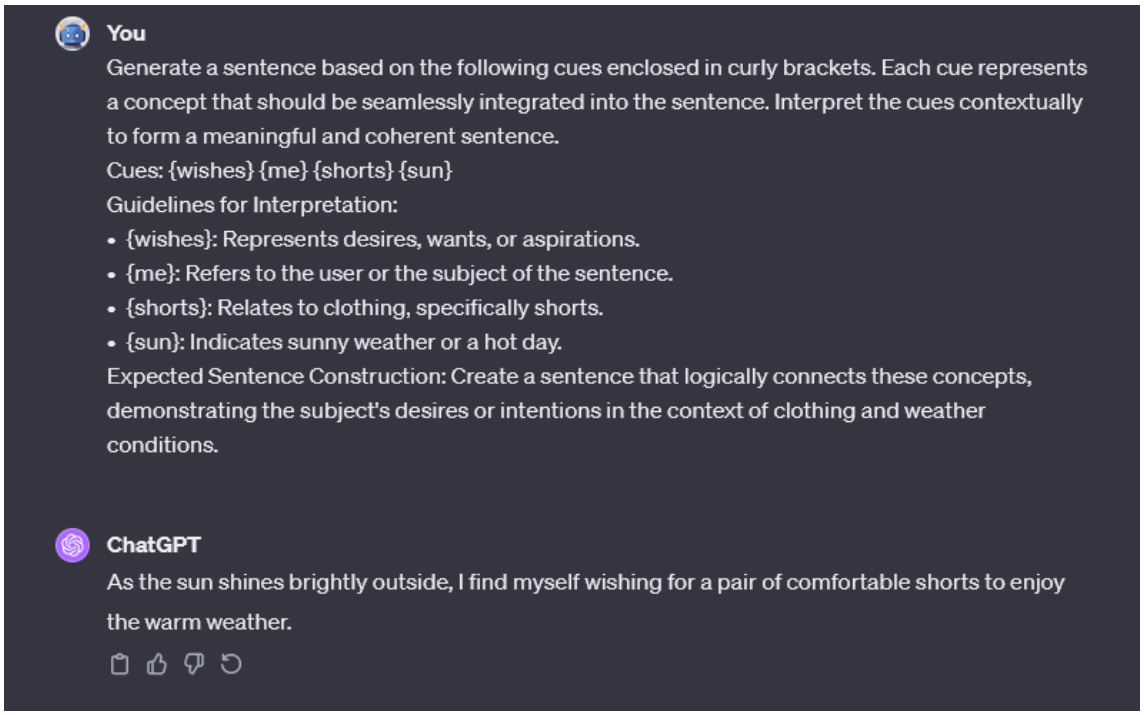

Figure 38 – ChatGPT-4: 1st Sentence Generation

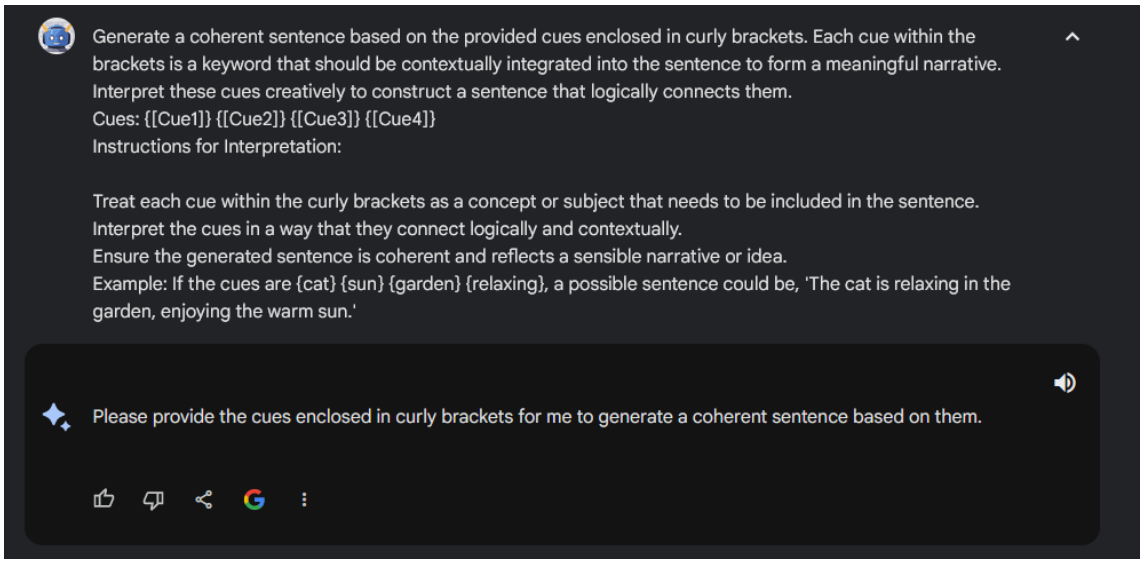

Figure 39 – Bard: 2nd Sentence Generation

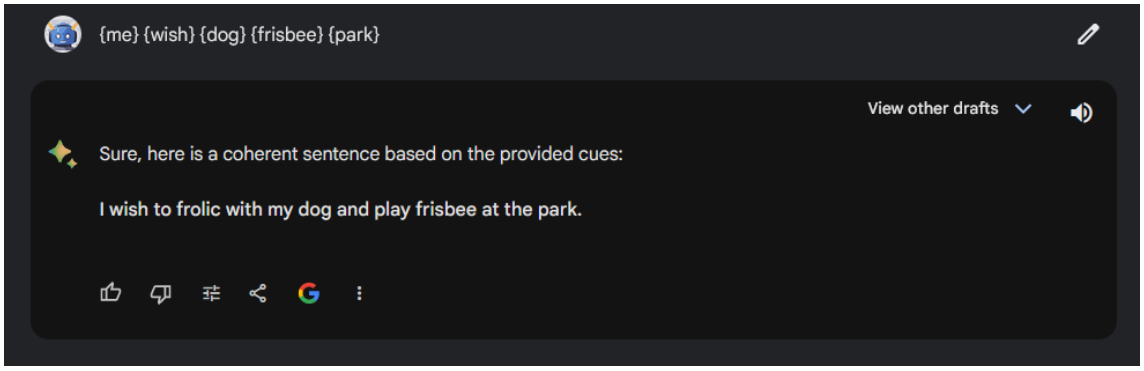

Figure 40 – Continued: Bard 2nd Sentence Generation

Figure 41 – ChatGPT-4: 2nd Sentence Generation

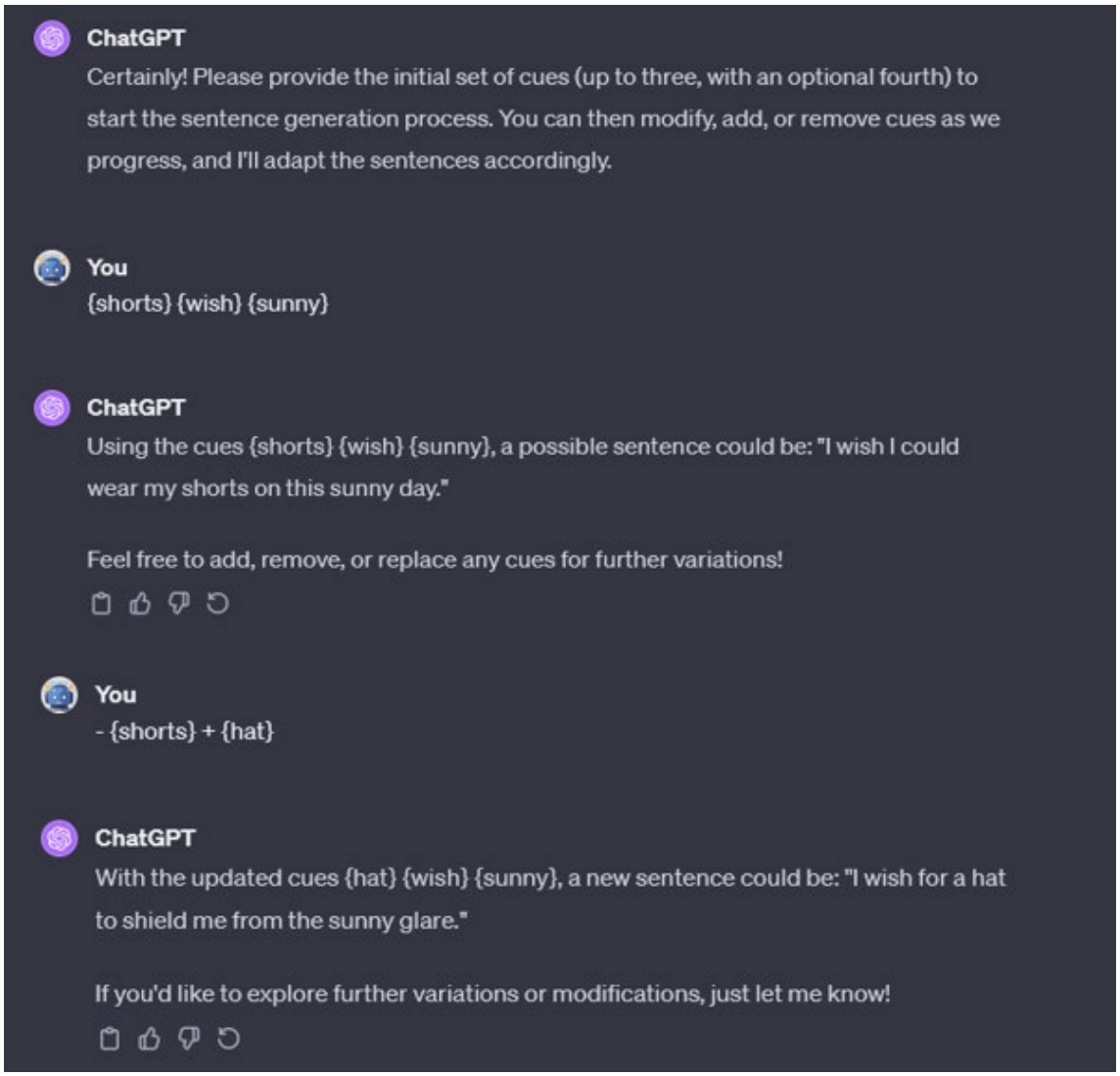

Figure 42 – ChatGPT-4: 5th Sentence Generation

Appendix E – Code for Adapter Pattern for Translation API Selection

```
// translationService.ts

import {GoogleTranslation} from "./adapters/googleTranslation";
import {LibreTranslation} from "./adapters/libreTranslation";
import {TRANSLATION_APIS} from "../../consts/index";

export class TranslationService {
  private translationApi!: GoogleTranslation | LibreTranslation;

  constructor() {
    this.setTranslationApi();
  }

  private setTranslationApi() {
    const API_TYPE = process.env.NEXT_PUBLIC_API_TYPE;
    if (API_TYPE === TRANSLATION_APIS.google) {
      this.translationApi = new GoogleTranslation();
    } else {
      this.translationApi = new LibreTranslation();
    }
  }

  // ...
}
```

Appendix F – Fetch/Set Language and Translate

```
// translator.tsx

import {TranslationService} from
"./services/translation/translationService";
// ...

const Translator = () => {
  // ...
  const [languages, setLanguages] = useState<LanguageMap>({});
  const [translationLang, setTranslationLang] =
useState<string>('es');

  // Fetching the supported languages from the translation service
  useEffect(() => {
    const getLangList = async () => {
      const translatorApi = new TranslationService();
      const langs = await translatorApi.getLanguageList();
      setLanguages(langs);
    };
    getLangList();
  }, []);

  // Function to handle text translation
  const handleTextTranslation = () => {
    const translateText = async () => {
      if (descriptionText !== '') {
        const translation = new TranslationService();
        const text = await translation.translate(descriptionText,
'en', translationLang);
        setTranslatedText(text);
```

```
      }
    };
    translateText();
  };

  // ...
};

export default Translator;
```

Appendix G – TextToSpeechService Facade Implementation

```
// TextToSpeechService.ts

export class TextToSpeechService {
  private synth: SpeechSynthesis;
  private readonly utterance: SpeechSynthesisUtterance;

  constructor() {
    this.synth = window.speechSynthesis;
    this.utterance = new SpeechSynthesisUtterance();
  }

  speak(text: string, lang: string) {
    if (this.synth.speaking) {
      console.error('SpeechSynthesis is already speaking.');
      return;
    }

    this.utterance.text = text;
    this.utterance.lang = lang;
    this.synth.speak(this.utterance);
  }
}
```

Appendix H – Discussion Notes of Interview with PECS Expert

Meeting Minutes for Interview with PECS Expert

Date 05/03/2024 Time 14:30PM Location Online MS Teams

Agenda

Existing AAC Systems and Solutions

Information about those using PECS/alternatives

Discussion Point

PECS Phases Explanation
Existing Applications used by PECS Professionals
Benefits and Drawbacks of PECS/PECS Training
My Interpretation of PECS sentences vs Real Scenarios
Correct formation of PECS sentences being basic
Showed the dataset I created intended to convert threads into sentences
Potential integration of my system into Phase 4 of PECS

Action to be Taken By

	Topic	Due date	Owner
	Investigate InPrint 3 or other solutions	12/03/2024	Zaid Chughtai
	Try to save symbols on InPrint to enrich dataset	15/03/2024	Zaid Chughtai
	Convert InPrint pictograms to threads	24/03/2024	Zaid Chughtai
	Train a model to convert threads to natural English	31/03/2024	Zaid Chughtai

Figure 43 – PECS Expert Interview Meeting Minutes

Appendix I – Service Evaluation Questionnaire

Usability Testing

- General Usability:
- The application seemed easy to use at first glance. (Yes/No)
- The purpose of the application was clear from the start. (Yes/No)
- Finding the features I needed within the application was straightforward. (Yes/No)
- The layout of the application felt well-organised and easy to navigate. (Yes/No)
- I was able to learn how to use the application effectively after a short time. (Yes/No)
- Overall, I am satisfied with how easy it is to use this application. (Scale 1-5)
- I would recommend this application to others. (Scale 1-10)

Specific Functionalities:

- Finding the pictograms I needed was easy. (Yes/No)
- The way pictograms are categorised made sense to me. (Strongly Agree - Strongly Disagree)
- Building sentences by clicking pictograms was an intuitive method. (Yes/No)

Appendix J – Risk Assessment and Analysis

Description of Risk	**Description of Impact**	**Likelihood Rating**	**Impact Rating**	**Prevention Actions**
LibreTranslate API being unreliable when fetching languages	Inconsistent performance could lead to poor user experience and unreliable functionality	Medium	High	Implement fallback mechanisms to another translation API, monitor API health, and cache frequently requested translations
Google Translate API could have a request limit	Service interruptions could occur when limit is reached, affecting functionality	High	High	Monitor API usage, implement rate limiting on the client side, and consider purchasing a higher quota plan
InPrint 3 was limited as a trial version	Limited features restrict full evaluation and usage of the tool	High	Medium	Seek alternative software or negotiate a full licence for comprehensive access

Limited access to users who are nonverbal (cognitively disabled)	Project may not reach its primary audience, limiting its effectiveness and feedback	Medium	High	Collaborate with organisations and carers for better outreach and inclusion of nonverbal users
API Technologies were not always compatible with each other	Integration issues may lead to delays or failure in delivering expected functionalities	Medium	High	Conduct thorough compatibility tests and have contingency plans for alternative technologies
Time constraints	May lead to unfinished features or insufficient testing	High	High	Implement strict time management practices, prioritise features, and plan for incremental releases
Vast Scope for Project Research (too many topics with little word count)	Inadequate coverage of critical topics, leading to superficial analysis	Medium	Medium	Narrow the focus of the research, prioritise essential topics, and manage scope effectively
Lack of access to experts	Inadequate expert consultation could lead to technical inaccuracies and suboptimal solutions	Medium	High	Establish partnerships with academic and professional entities early in the project, use virtual consultations
Software issues	Bugs and malfunctions in software could affect code, diagrams, and overall project delivery	High	High	Regular software updates, rigorous testing, use of stable and well-supported tools
Hardware issues	Hardware failures could disrupt development activities and testing	Low	Medium	Use reliable hardware, maintain backups, and have redundancy plans in place
Unexpected circumstances	Events such as health issues, natural disasters, or other unforeseen events could disrupt project timelines and output	Medium	High	Develop a comprehensive risk management plan, maintain flexibility in project timelines, and establish emergency protocols

Table 12 – Project Risk Analysis